



Source Code For Phase Contrast Magnetic Resonance Flow Measurement System

CANVAS Group

Authors: Meide Zhao, PhD, mzhao@uic.edu

September, 1999

```

////////////////////////////////////
// COPYRIGHT September 1999 University of Illinois at Chicago
// Authors: Meide Zhao, Neurosurgery Department
//           University of Illinois at Chicago
//           Email: mzhao@uic.edu
// CANVAS Medical Imaging System and
// Phase Contrast MR Flow Measurement System Using 3D Localization
////////////////////////////////////

#include <Vk/VkApp.h>
// Headers for window classes used in this program
#include "VkwindowMainWindow.h"
//----- Start editable code block: headers and declarations
//----- End editable code block: headers and declarations
// Fallback resources

static char *fallbackResources[] = {
    "useSchemes:    all",
    "sgiMode:       true",
    "useEnhancedFSB: true",
    "keyboardFocusPolicy: explicit",
    //----- Start editable code block: fallbacks

    //----- End editable code block: fallbacks
    NULL
};

void main ( int argc, char **argv )
{
    extern void InitEZ(void);

    InitEZ(); // Only need to force bind EZ library           // for
    Fix+Continue

    //----- Start editable code block: main initialization
    //----- End editable code block: main initialization

    VkApp::setFallbacks(fallbackResources);

    VkApp      *app;

    // Create an application object

    app = new VkApp("Cmis", &argc, argv);

    //----- Start editable code block: post init
    //----- End editable code block: post init

    // Create the top level windows
    VkSimpleWindow *vkwindow;
    if(argc == 1)
        vkwindow = new VkwindowMainWindow("vkwindow");
    else if(argc == 2)
        vkwindow = new VkwindowMainWindow(argv[1]);

    vkwindow->show();
}

```



```
vkwindow -> setTitle("CMIS (Cancer Medical Imaging System)
```

```
//---- End editable code block: event loop
```

```
app->run ();
```

```
}
```

```
//---- Start editable code block: End of generated code
```

```
-//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for DrawingAreaUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "DrawingAreaUI.h" // Generated header file for this class

#include <Xm/DrawingA.h>
#include <Vk/VkResource.h>
//---- Start editable code block: headers and declarations

#include <Xm/ScrolledW.h>

//---- End editable code block: headers and declarations


// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String DrawingAreaUI::_defaultDrawingAreaUIResources[] = {

    //---- Start editable code block: DrawingAreaUI Default Resources

    //---- End editable code block: DrawingAreaUI Default Resources

    (char*)NULL
};

DrawingAreaUI::DrawingAreaUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: DrawingArea constructor 2

    //---- End editable code block: DrawingArea constructor 2

} // End Constructor

```

[illegible]

```

{
    _baseWidget2 = _drawingArea = XtVaCreateWidget ( _parent,
                                                    xmDrawingAreaWidgetClass,
                                                    parent,
                                                    (XtPointer) NULL );

    _baseWidget = _baseWidget2;
}

// install a callback to guard against unexpected widget destruction
installDestroyHandler();

XtAddCallback ( _baseWidget,
                XmNresizeCallback,
                &DrawingAreaUI::resizeCallback,
                (XtPointer) this );

XtAddCallback ( _baseWidget,
                XmNexposeCallback,
                &DrawingAreaUI::exposeCallback,
                (XtPointer) this );

XtAddCallback ( _baseWidget,
                XmNinputCallback,
                &DrawingAreaUI::inputCallback,
                (XtPointer) this );

XtAddEventHandler( _baseWidget,
                  PointerMotionMask, FALSE,
                  &DrawingAreaUI::motion,
                  (XtPointer) this);

//----- Start editable code block: DrawingAreaUI create

//----- End editable code block: DrawingAreaUI create
}

const char * DrawingAreaUI::className()
{
    return ("DrawingAreaUI");
} // End className()

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

void DrawingAreaUI::exposeCallback ( Widget w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    DrawingAreaUI* obj = ( DrawingAreaUI * ) clientData;
    obj->expose ( w, callData );
}

void DrawingAreaUI::inputCallback ( Widget w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    DrawingAreaUI* obj = ( DrawingAreaUI * ) clientData;
    obj->input ( w, callData );
}

```

```

void DrawingAreaUI::resizeCallback ( Widget w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    DrawingAreaUI* obj = ( DrawingAreaUI * ) clientData;
    obj->resize ( w, callData );
}

void DrawingAreaUI::motion(Widget w, XtPointer clientData,
                           XEvent *event, Boolean *flag)
{
    DrawingAreaUI* obj = ( DrawingAreaUI * ) clientData;
    obj->motion ( w, event );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void DrawingAreaUI::expose ( Widget, XtPointer )
{
    // This virtual function is called from exposeCallback.
    // This function is normally overridden by a derived class.
}

void DrawingAreaUI::input ( Widget, XtPointer )
{
    // This virtual function is called from inputCallback.
    // This function is normally overridden by a derived class.
}

void DrawingAreaUI::resize ( Widget, XtPointer )
{
    // This virtual function is called from resizeCallback.
    // This function is normally overridden by a derived class.
}

void DrawingAreaUI::motion ( Widget, XEvent * )
{
    // This virtual function is called from resizeCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for DrawingArea
//
// This file is generated by RapidApp 1.2
//
// This class is derived from DrawingAreaUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "DrawingArea.h"
#include <Vk/VkEZ.h>
#include <Xm/DrawingA.h>
#include <Vk/VkResource.h>
#include <Vk/VkSimpleWindow.h>

extern void VkUnimplemented ( Widget, const char * );

///----- Start editable code block: headers and declarations

#include <stdio.h>
#include <fstream.h>

///----- End editable code block: headers and declarations

///----- DrawingArea Constructor

DrawingArea::DrawingArea(const char *name) :
    DrawingAreaUI(name)
{
    // This constructor calls DrawingAreaUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    ///----- Start editable code block: DrawingArea constructor 2

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    ///----- End editable code block: DrawingArea constructor 2

} // End Constructor

DrawingArea::DrawingArea(const char *name, Widget parent, int flag) :
    DrawingAreaUI(name, parent, flag)

```

```

{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    MakeColormap(parent);
    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    //----- End editable code block: DrawingArea constructor

}    // End Constructor

DrawingArea::DrawingArea(int w, int h, const char *name,
    Widget parent, int flag) : DrawingAreaUI(name, parent, flag)
{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    width = w;
    height = h;

    MakeColormap(parent);

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    printf(" DrawingArea is done \n");

    //----- End editable code block: DrawingArea constructor

}    // End Constructor

DrawingArea::DrawingArea(int w, int h, unsigned char **grayimg, const char *name,
    Widget parent, int flag) : DrawingAreaUI(name, parent, flag)
{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    width = w;
    height = h;

    MakeColormap(parent);

    create_pixmap(w, h, grayimg);

    //----- End editable code block: DrawingArea constructor

```

```
} // End Constructor
```

```
DrawingArea::~DrawingArea()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: DrawingArea destructor

    clear_memory();

    //---- End editable code block: DrawingArea destructor

} // End Destructor
```

```
void DrawingArea::clear_memory()
```

```
{
    if (_pixmap != NULL)
    { XFreePixmap(XtDisplay(baseWidget()), _pixmap); _pixmap = NULL; }
    if (_gc != NULL)
    { XtReleaseGC(baseWidget(), _gc); _gc = NULL; }
    if (_ximage != NULL)
    { XDestroyImage(_ximage); _ximage = NULL; }

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;
}
```

```
void DrawingArea::set_imgdata(unsigned char **grayimg)
```

```
{
    clear_memory();
    create_pixmap(width, height, grayimg);
}
```

```
void DrawingArea::set_imgdata(int w,int h, unsigned char **grayimg)
```

```
{
    clear_memory();

    width = w;
    height = h;

    create_pixmap(w, h, grayimg);
}
```

```
void DrawingArea::set_imgdata(ColorImage *img)
```

```
{
    if(img != NULL)
    {
        clear_memory();
        create_pixmap(width, height, img->red, img->green, img->blue);
    }
}
```

```
void DrawingArea::set_imgdata(int w,int h, ColorImage *img)
```

```
{
    if(img != NULL)
    {
        clear_memory();
    }
}
```



```

        width = w;
        height = h;
        create_pixmap(width, height, img->red, img->green, img->blue);
    }
}

Pixmap DrawingArea::get_pixmap1(ColorImage *img)
{
    if(img != NULL)
    {
        return get_pixmap2(width, height, img->red, img->green, img->blue);
    }
    return NULL;
}

const char * DrawingArea::className() // classname
{
    return ("DrawingArea");
} // End className()

void DrawingArea::MakeColormap(Widget w)
{
    Pixel      cells[128];
    Colormap   cmap;
    XColor     color[128];

    Display *dpy = XtDisplay(w);
    Screen *screen = XtScreen(w);

    cmap = DefaultColormapOfScreen(screen);

    XInstallColormap(dpy,  cmap);

    if(XDefaultDepthOfScreen(screen) > 8)
    {
        _depth = 24;
        printf(" \n\n MakeColormap   Depth %d \n\n", _depth);
        return;
    }
    else _depth = 8;

    printf(" \n\n MakeColormap   Depth %d \n\n", _depth);

    if( XAllocColorCells(dpy,cmap,TRUE,NULL,0,cells,128) )
        cout<<" XAllocColorCells is done !"<<endl;

    _offset=cells[0];

    for(int i=0;i<128;i++)
    {
        // FOR COLOR PIXMAPS

        color[i].red=(i & 0x70)*585;
        color[i].green=(i & 0xC)*5461;
        color[i].blue=(i & 0x3)*21845;

        // FOR MONOCHROME PIXMAPS
        /*
        color[i].red = color[i].green = color[i].blue= i*516;
        */
        color[i].flags = DoRed | DoGreen | DoBlue;
        color[i].pixel = cells[i];
    }
}

```

```

8      /* bit map pad */,
0      /* bytes per line (self calculate) */
);

```

11

```

return(xi);
}

void DrawingArea::set_Origin(int x, int y)
{
    XtVaSetValues ( baseWidget(),
                    XmNx, x,
                    XmNy, y,
                    (XtPointer) NULL );
    XtRealizeWidget(baseWidget());
}

void DrawingArea::display(int x, int y)
{
    //show();
    XtVaSetValues ( baseWidget(),
                    XmNx, x,
                    XmNy, y,
                    XmNwidth, width,
                    XmNheight, height,
                    (XtPointer) NULL );
}

void DrawingArea::display(int x, int y, int w, int h)
{
    //show();
    XtVaSetValues ( baseWidget2(),
                    XmNx, x,
                    XmNy, y,
                    XmNwidth, w,
                    XmNheight, h,
                    (XtPointer) NULL );
    XtVaSetValues ( baseWidget(),
                    XmNwidth, width,
                    XmNheight, height,
                    (XtPointer) NULL );
}

void DrawingArea::copyArea(int x, int y, int w, int h)
{
    XCopyArea(XtDisplay(baseWidget()), _pixmap, XtWindow(baseWidget()), _gc, x, y, w, h, x
}

void DrawingArea::display()
{
    if(_pixmap != NULL)
    {
        XtVaSetValues ( baseWidget(),
                        XmNwidth, width,
                        XmNheight, height,
                        (XtPointer) NULL );
        XCopyArea(XtDisplay(baseWidget()), _pixmap, XtWindow(baseWidget()), _gc, 0, 0, width
    }
}

void DrawingArea::expose ( Widget wid, XtPointer callData )
{
    //---- Start editable code block: DrawingArea expose

    XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::expose is implemented:

```

```

//::VkUnimplemented ( "DrawingArea::expose" );

display();

//----- End editable code block: DrawingArea expose
} // End DrawingArea::expose()

void DrawingArea::input ( Widget w, XtPointer callData )
{
    //----- Start editable code block: DrawingArea input

    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::input is implemented:
    //::VkUnimplemented ( w, "DrawingArea::input" );

    printf(" Input \n");

    printf(" %d\n",cb->event->type);

    if (cb->event->type == ButtonPress)
    {
        if (cb->event->xbutton.button == Button3)
            printf("Button3\n");
        else if (cb->event->xbutton.button == Button2)
            printf("Button2\n");
        else if (cb->event->xbutton.button == Button1)
        {
            printf("Button1\n");
        }
    }
    else if (cb->event->type == ButtonRelease)
    {
        if (cb->event->xbutton.button == Button3)
        {
            printf(" R Button3\n");
        }
        else if (cb->event->xbutton.button == Button2)
        {
            printf(" R Button2\n");
        }
        else if (cb->event->xbutton.button == Button1)
        {
            printf(" R Button1\n");
        }
    }

    //----- End editable code block: DrawingArea input
} // End DrawingArea::input()

void DrawingArea::resize ( Widget w, XtPointer callData )
{
    //----- Start editable code block: DrawingArea resize

    XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::resize is implemented:

```

```
//---- End editable code block: DrawingArea resize
} // End DrawingArea::resize()

void DrawingArea::motion ( Widget w, XEvent *event )
{
    //---- Start editable code block: DrawingArea resize
    //XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;
    //--- Comment out the following line when DrawingArea::resize is implemented:
    //::VkUnimplemented ( w, "DrawingArea::resize" );

    //printf(" motion \n");
    int xposition = event->xmotion.x;
    int yposition = event->xmotion.y;
    //printf("\nX: %3d   Y: %3d ", xposition, yposition);

    //---- End editable code block: DrawingArea resize
} // End DrawingArea::resize()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *DrawingArea::CreateDrawingArea( const char *name, Widget parent, int flag
{
    VkComponent *obj = new DrawingArea ( name, parent , flag);
    return ( obj );
} // End CreateDrawingArea

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *DrawingArea::RegisterDrawingAreaInterface()
{
    // This structure registers information about this class
```

```

// that allows RapidA to create and manipulate an instance.
// Each entry provides a resource name that will appear in the
// resource manager palette when an instance of this class is
// selected, the name of the member function as a string,
// the type of the single argument to this function, and an
// optional argument indicating the class that defines this function.
// All member functions must have the form
//
//     void memberFunction ( Type );
//
// where "Type" is one of:
//     const char *      (Use XmRString)
//     Boolean           (Use XmRBoolean)
//     int               (Use XmRInt)
//     float             (Use XmRFloat)
//     No argument       (Use VkrNoArg or "NoArg")
//     A filename        (Use VkrFilename or "Filename")
//     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
//     A callback        (Use XmRCallback)

static InterfaceMap map[] = {
//---- Start editable code block: DrawingAreaUI resource table

    { "resourceName", "setAttribute", XmRString },
//---- End editable code block: DrawingAreaUI resource table
    { NULL }, // MUST be NULL terminated
};

return map;
} // End RegisterDrawingAreaInterface()

//---- End of generated code

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for DrawingArea
//
// This file is generated by RapidApp 1.2
//
// This class is derived from DrawingAreaUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "DrawingArea.h"
#include <Vk/VkEZ.h>
#include <Xm/DrawingA.h>
#include <Vk/VkResource.h>
#include <Vk/VkSimpleWindow.h>

extern void VkUnimplemented ( Widget, const char * );

///----- Start editable code block: headers and declarations

#include <stdio.h>
#include <fstream.h>

///----- End editable code block: headers and declarations

///----- DrawingArea Constructor

DrawingArea::DrawingArea(const char *name) :
    DrawingAreaUI(name)
{
    // This constructor calls DrawingAreaUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    ///----- Start editable code block: DrawingArea constructor 2

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    ///----- End editable code block: DrawingArea constructor 2

} // End Constructor

DrawingArea::DrawingArea(const char *name, Widget parent, int flag) :
    DrawingAreaUI(name, parent, flag)

```

```

{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    MakeColormap(parent);
    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    //----- End editable code block: DrawingArea constructor

}    // End Constructor

DrawingArea::DrawingArea(int w, int h, const char *name,
    Widget parent, int flag) : DrawingAreaUI(name, parent, flag)
{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    width = w;
    height = h;

    MakeColormap(parent);

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;

    printf(" DrawingArea is done \n");

    //----- End editable code block: DrawingArea constructor

}    // End Constructor

DrawingArea::DrawingArea(int w, int h, unsigned char **grayimg, const char *name,
    Widget parent, int flag) : DrawingAreaUI(name, parent, flag)
{
    // This constructor calls DrawingAreaUI(parent, name)
    // which calls DrawingAreaUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: DrawingArea constructor

    width = w;
    height = h;

    MakeColormap(parent);

    create_pixmap(w, h, grayimg);

    //----- End editable code block: DrawingArea constructor

```

```
} // End Constructor
```

```
DrawingArea::~DrawingArea()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: DrawingArea destructor

    clear_memory();

    //---- End editable code block: DrawingArea destructor
}
```

```
// End Destructor
```

```
void DrawingArea::clear_memory()
```

```
{
    if (_pixmap != NULL)
    { XFreePixmap(XtDisplay(baseWidget()), _pixmap); _pixmap = NULL; }
    if (_gc != NULL)
    { XtReleaseGC(baseWidget(), _gc); _gc = NULL; }
    if (_ximage != NULL)
    { XDestroyImage(_ximage); _ximage = NULL; }

    _pixmap = NULL;
    _gc = NULL;
    _ximage = NULL;
}
```

```
void DrawingArea::set_imgdata(unsigned char **grayimg)
```

```
{
    clear_memory();
    create_pixmap(width, height, grayimg);
}
```

```
void DrawingArea::set_imgdata(int w, int h, unsigned char **grayimg)
```

```
{
    clear_memory();

    width = w;
    height = h;

    create_pixmap(w, h, grayimg);
}
```

```
void DrawingArea::set_imgdata(ColorImage *img)
```

```
{
    if (img != NULL)
    {
        clear_memory();
        create_pixmap(width, height, img->red, img->green, img->blue);
    }
}
```

```
void DrawingArea::set_imgdata(int w, int h, ColorImage *img)
```

```
{
    if (img != NULL)
    {
        clear_memory();
    }
}
```


}

Pixmap DrawingArea::get_pixmap2(int w, int h, unsigned char **r,
unsigned char **g, unsigned char **b)

{

if(r == NULL && g == NULL && b == NULL)
{
return NULL;
}

Widget wid = baseWidget();
Display *dpy = XtDisplay(wid);
Screen *screen = XtScreen(wid);

GC gc = DefaultGCOfScreen(screen);

unsigned char *imgdata;
if(r != NULL && g == NULL && b == NULL)
imgdata = toXdata(0, w, h, r);
else if(r != NULL && g != NULL && b != NULL)
imgdata = toXdata(1, w, h, r, g, b);

XImage *ximage = img2XImage(dpy, screen, imgdata, w, h, _depth);

Pixmap pixmap = XCreatePixmap(dpy, RootWindowOfScreen(screen), w, h, _depth);

XPutImage(dpy, pixmap, gc, ximage, 0, 0, 0, 0, w, h);

delete imgdata;
XDestroyImage(ximage);
XtReleaseGC(baseWidget(), gc);

return pixmap;

}

unsigned char *DrawingArea::toXdata(int bw, int w, int h, unsigned char **grayimg)

{

unsigned char *img;
float tmp;
int pos;

if(_depth > 8) img = new unsigned char[w*h*4];
else img = new unsigned char[w*h];

for (int i=0; i<h; i++)
for(int j=0; j<w; j++)
{

tmp = grayimg[i][j];

if(_depth > 8) pos = (i*w + j)*4;
else pos = i*w + j;

setXData(bw, _depth, _offset, (int)tmp, (int)tmp, (int)tmp, pos, img);

}

return img;

}

unsigned char *DrawingArea::toXdata(int bw, int w, int h, unsigned char **r,
unsigned char **g, unsigned char **b)

{

unsigned char *img;
float tmp;

```

8      /* bit map pad */,
0      /* bytes per line (self calculate) */
);

```

19

```

return(xi);
}

```

```

void DrawingArea::set_Origin(int x, int y)
{
    XtVaSetValues ( baseWidget(),
                    XmNx, x,
                    XmNy, y,
                    (XtPointer) NULL );
    XtRealizeWidget(baseWidget());
}

```

```

void DrawingArea::display(int x, int y)
{
    //show();
    XtVaSetValues ( baseWidget(),
                    XmNx, x,
                    XmNy, y,
                    XmNwidth, width,
                    XmNheight, height,
                    (XtPointer) NULL );
}

```

```

void DrawingArea::display(int x, int y, int w, int h)
{
    //show();
    XtVaSetValues ( baseWidget2(),
                    XmNx, x,
                    XmNy, y,
                    XmNwidth, w,
                    XmNheight, h,
                    (XtPointer) NULL );
    XtVaSetValues ( baseWidget(),
                    XmNwidth, width,
                    XmNheight, height,
                    (XtPointer) NULL );
}

```

```

void DrawingArea::copyArea(int x, int y, int w, int h)
{
    XCopyArea(XtDisplay(baseWidget()), _pixmap, XtWindow(baseWidget()), _gc, x, y, w, h);
}

```

```

void DrawingArea::display()
{
    if(_pixmap != NULL)
    {
        XtVaSetValues ( baseWidget(),
                        XmNwidth, width,
                        XmNheight, height,
                        (XtPointer) NULL );
        XCopyArea(XtDisplay(baseWidget()), _pixmap, XtWindow(baseWidget()), _gc, 0, 0, width, height);
    }
}

```

```

void DrawingArea::expose ( Widget wid, XtPointer callData )
{
    //---- Start editable code block: DrawingArea expose

    XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::expose is implemented:

```

```

//::VkUnimplemented "DrawingArea::expose" );

display();

//---- End editable code block: DrawingArea expose
} // End DrawingArea::expose()

void DrawingArea::input ( Widget w, XtPointer callData )
{
    //---- Start editable code block: DrawingArea input

    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::input is implemented:
    //::VkUnimplemented ( w, "DrawingArea::input" );

    printf(" Input \n");

    printf(" %d\n",cb->event->type);

    if (cb->event->type == ButtonPress)
    {
        if (cb->event->xbutton.button == Button3)
            printf("Button3\n");
        else if (cb->event->xbutton.button == Button2)
            printf("Button2\n");
        else if (cb->event->xbutton.button == Button1)
        {
            printf("Button1\n");
        }
    }
    else if (cb->event->type == ButtonRelease)
    {
        if (cb->event->xbutton.button == Button3)
        {
            printf(" R Button3\n");
        }
        else if (cb->event->xbutton.button == Button2)
        {
            printf(" R Button2\n");
        }
        else if (cb->event->xbutton.button == Button1)
        {
            printf(" R Button1\n");
        }
    }

    //---- End editable code block: DrawingArea input
} // End DrawingArea::input()

void DrawingArea::resize ( Widget w, XtPointer callData )
{
    //---- Start editable code block: DrawingArea resize

    XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::resize is implemented:

```

```

//----- End editable code block: DrawingArea resize
} // End DrawingArea::resize()

void DrawingArea::motion ( Widget w, XEvent *event )
{
    //----- Start editable code block: DrawingArea resize
    //XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;
    //--- Comment out the following line when DrawingArea::resize is implemented:
    //::VkUnimplemented ( w, "DrawingArea::resize" );

    //printf(" motion \n");
    int xposition = event->xmotion.x;
    int yposition = event->xmotion.y;
    //printf("\nX: %3d   Y: %3d ", xposition, yposition);

    //----- End editable code block: DrawingArea resize
} // End DrawingArea::resize()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *DrawingArea::CreateDrawingArea( const char *name, Widget parent, int flag
{
    VkComponent *obj = new DrawingArea ( name, parent , flag);
    return ( obj );
} // End CreateDrawingArea

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *DrawingArea::RegisterDrawingAreaInterface()
{
    // This structure registers information about this class

```

```

// that allows RapidApp to create and manipulate an instance.
// Each entry provides a resource name that will appear in the
// resource manager palette when an instance of this class is
// selected, the name of the member function as a string,
// the type of the single argument to this function, and an
// optional argument indicating the class that defines this function.
// All member functions must have the form
//
//      void memberFunction ( Type );
//
// where "Type" is one of:
//      const char *      (Use XmRString)
//      Boolean           (Use XmRBoolean)
//      int               (Use XmRInt)
//      float             (Use XmRFloat)
//      No argument       (Use VkrNoArg or "NoArg")
//      A filename        (Use VkrFilename or "Filename")
//      An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
//      A callback        (Use XmRCallback)

static InterfaceMap map[] = {
//---- Start editable code block: DrawingAreaUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: DrawingAreaUI resource table
    { NULL }, // MUST be NULL terminated
};

return map;
} // End RegisterDrawingAreaInterface()

//---- End of generated code

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

#include "ImgAlloc.h"

#include <stdio.h>

#include <iostream.h>

```
MedDrawingArea::MedDrawingArea(const char *name, Widget parent, int flag) :
    DrawingArea(name, parent, flag)
```

```
{
    _orgImg = NULL;
    _zoomImg = NULL;
    _zoom = 1;
    _winCenter = 0;
    _winWidth = 0;
    _button1Pressed = FALSE;
    _flowDir = 0;
    _cimg = NULL;
    _cimg2 = NULL;
    _minFlow = 0;
    _maxFlow = 0;
}
```

```
MedDrawingArea::~~MedDrawingArea()
```

```
{
    Utility_Vision *u = new Utility_Vision();
    if(_zoomImg != NULL)
        u -> freeShimg(_zoomImg);
    delete u;
    remove_cimg();
}
```

```
void MedDrawingArea::remove_cimg()
```

```
{
    Utility_Vision *u = new Utility_Vision();
    if(_cimg != NULL)
    {
        u -> freeCImg(_cimg);
        _cimg = NULL;
    }

    if(_cimg2 != NULL)
    {
        u -> freeCImg(_cimg2);
        _cimg2 = NULL;
    }
    delete u;
}
```

```
void MedDrawingArea::create_cimg2()
```

```
{
    int w = get_width();
    int h = get_height();

    _cimg2 = new ColorImage;
    _cimg2 -> red = alloc_img(w, h);
    _cimg2 -> green = alloc_img(w, h);
    _cimg2 -> blue = alloc_img(w, h);

    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
    {
        //highlight(60.0, (_cimg -> red)[i][j], (_cimg -> green)[i][j], (_cimg -> blue)
        // &((_cimg2 -> red)[i][j]), &((_cimg2 -> green)[i][j]), &((_cimg2 -> blue)[i]
        _cimg2 -> red[i][j] = _cimg -> red[i][j];
        _cimg2 -> green[i][j] = _cimg -> green[i][j];
    }
}
```

```

        _cimg2 -> blue[i][j] = _cimg -> blue[i][j];
    }
}

void MedDrawingArea::set(int w, int h, short **img, int visual_method,
    int scale_method, float zoom, float winCenter, float winWidth, int flowDir)
{
    int w2, h2;
    Utility_Vision *u = new Utility_Vision();

    _orgImg = img;
    _orgWidth = w;
    _orgHeight = h;

    _visual_method = visual_method;
    _scale_method = scale_method;
    _zoom = zoom;
    _winCenter = winCenter;
    _winWidth = winWidth;

    _flowDir = flowDir;
    //printf(" MedDrawingArea :: FlowDir %d\n", _flowDir);

    //printf(" MedDrawingArea :: Org: %d %d zoom: %f\n", w, h, zoom);

    if(_zoomImg != NULL) u -> freeShimg(_zoomImg);
    _zoomImg = u -> scale_img(scale_method, w, h, img, zoom, &w2, &h2);

    remove_cimg();

    _cimg = toVisual(visual_method, w2, h2, _zoomImg, winCenter, winWidth);

    DrawingArea::set_imgdata(w2, h2, _cimg);

    delete u;
}

void MedDrawingArea::setData(int w, int h, short **img, int visual_method,
    int scale_method, float zoom, float winCenter, float winWidth, int flowDir)
{
    int w2, h2;
    Utility_Vision *u = new Utility_Vision();

    _orgImg = img;
    _orgWidth = w;
    _orgHeight = h;

    _visual_method = visual_method;
    _scale_method = scale_method;
    _zoom = zoom;
    _winCenter = winCenter;
    _winWidth = winWidth;

    _flowDir = flowDir;

    if(_zoomImg != NULL) u -> freeShimg(_zoomImg);
    _zoomImg = u -> scale_img(scale_method, w, h, img, zoom, &w2, &h2);

    delete u;
}

Pixmap MedDrawingArea::get_pixmap(short **img)
{
    int w2, h2;

```

```

        if(_zoomImg[i][j] >= _minI && _zoomImg[i][j] <= maxI)
        {
            if(_minFlow > _zoomImg[i][j]) _minFlow = _zoomImg[i][j];
            if(_maxFlow < _zoomImg[i][j]) _maxFlow = _zoomImg[i][j];
        }
    }
}

```

```

Boolean MedDrawingArea::update(float winCenter, float winWidth)

```

```

{
    if(_winCenter == winCenter && _winWidth == winWidth) return FALSE;
    _winCenter = winCenter;
    _winWidth = winWidth;

    Utility_Vision *u = new Utility_Vision();

    int w = get_width();
    int h = get_height();

    remove_cimg();

    _cimg = toVisual(_visual_method, w, h, _zoomImg, winCenter, winWidth);

    DrawingArea::set_imgdata(_cimg);

    DrawingArea::display();

    delete u;
    return TRUE;
}

```

```

Boolean MedDrawingArea::update(int scale_method)

```

```

{
    if(_scale_method == scale_method) return FALSE;
    _scale_method = scale_method;

    int w2, h2;
    Utility_Vision *u = new Utility_Vision();

    if(_zoomImg != NULL) u -> freeShimg(_zoomImg);

    _zoomImg = u -> scale_img(scale_method, _orgWidth, _orgHeight, _orgImg, _zoom, &w2, &h2);

    remove_cimg();

    _cimg = toVisual(_visual_method, w2, h2, _zoomImg, _winCenter, _winWidth);

    DrawingArea::set_imgdata(w2, h2, _cimg);

    DrawingArea::display();

    delete u;
    return TRUE;
}

```

```

Boolean MedDrawingArea::update(float zoom)

```

```

{
    if(_zoom == zoom) return FALSE;
    _zoom = zoom;

    int w2, h2;
    Utility_Vision *u = new Utility_Vision();

    printf(" zoom = %f\n", zoom);

    if(_zoomImg != NULL) u -> freeShimg(_zoomImg);
}

```



```
_zoomImg = u -> scale_img(_scale_method, _orgWidth, _orgHeight, _orgImg, zoom, &w2, &h2, 26);
```

```
remove_cimg();
```

```
_cimg = toVisual(_visual_method, w2, h2, _zoomImg, _winCenter, _winWidth);
```

```
DrawingArea::set_imgdata(w2, h2, _cimg);
```

```
DrawingArea::display();
```

```
delete u;
```

```
return TRUE;
```

```
}
```

```
void MedDrawingArea::expose(Widget w, XtPointer callData)
```

```
{
```

```
    DrawingArea::display();
```

```
}
```

```
void MedDrawingArea::input(Widget w, XtPointer callData)
```

```
{
```

```
    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;
```

```
    int xpos = cb->event->xmotion.x;
```

```
    int ypos = cb->event->xmotion.y;
```

```
    //printf(" GE:  %d\n",cb->event->type);
```

```
    if (cb->event->type == ButtonPress)
```

```
    {
```

```
        if (cb->event->xbutton.button == Button3)
```

```
            printf("Button3\n");
```

```
        else if (cb->event->xbutton.button == Button2)
```

```
            printf("Button2\n");
```

```
        else if (cb->event->xbutton.button == Button1)
```

```
        {
```

```
            printf("Button1\n");
```

```
            _button1Pressed = TRUE;
```

```
        }
```

```
    }
```

```
    else if (cb->event->type == ButtonRelease)
```

```
    {
```

```
        if (cb->event->xbutton.button == Button3)
```

```
        {
```

```
            printf(" R Button3\n");
```

```
        }
```

```
        else if (cb->event->xbutton.button == Button2)
```

```
        {
```

```
            printf(" R Button2\n");
```

```
        }
```

```
        else if (cb->event->xbutton.button == Button1)
```

```
        {
```

```
            printf(" R Button1\n");
```

```
            _button1Pressed = FALSE;
```

```
        }
```

```
    }
```

```
}
```

```
void MedDrawingArea::motion ( Widget w, XEvent *event )
```

```
{
```

```
    //---- Start editable code block: DrawingArea resize
```

```
    //XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;
```

```
    //--- Comment out the following line when DrawingArea::resize is implemented:
```

```

//::VkUnimplemented ( "DrawingArea::resize" );

int xpos = event->xmotion.x;
int ypos = event->xmotion.y;

//---- End editable code block: DrawingArea resize
} // End MedDrawingArea::resize()

ColorImage *MedDrawingArea::toVisual(int visual_method, int w, int h, short **shimg, fl
{
    ColorImage *cimg;

    Utility_Vision *uv = new Utility_Vision();
    if(visual_method == VISUAL_COLOR && _flowDir == -1)
        cimg = uv -> toColor2(w, h, shimg, p1, p2);
    else
        cimg = uv -> toVisual(visual_method, w, h, shimg, p1, p2);
    delete uv;

    return cimg;
}

float **MedDrawingArea::getFloatImg()
{
    int w = get_width();
    int h = get_height();

    float **fimg = (float **)alloc_fimg(w, h);

    for(int i=0; i<h; i++)
        for(int j=0; j<w; j++)
            fimg[i][j] = (float)_zoomImg[i][j];

    return fimg;
}

```

```

////////////////////////////////////
// ROIMedDrawingArea.c++
////////////////////////////////////
#include "ROIMedDrawingArea.h"

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <X11/cursorfont.h>

#include "Utility_Math.h"
#include "Utility_Vision.h"
#include "Utility_Widget.h"
#include "ImgAlloc.h"

#include "Rectangle.h"
#include "Ellipse.h"
#include "Polygon.h"
#include "FreeHand.h"
#include "BbDetail.h"
#include "Bb3DLocalizer.h"

ROIMedDrawingArea::ROIMedDrawingArea(const char *name, Widget parent, int flag)
: MedDrawingArea(name,parent,flag)
{
    _ROI = NULL;
    _roi_color = COLOR_RED;
    _button2Pressed = FALSE;
}

ROIMedDrawingArea::~ROIMedDrawingArea()
{
    if(_ROI != NULL) delete _ROI;
}

unsigned char **ROIMedDrawingArea::get_mask()
{
    int w = get_width();
    int h = get_height();

    printf(" get_mask  %d %d\n", w, h);

    Utility_Widget *u = new Utility_Widget();
    unsigned char **area = u -> get_mask(baseWidget(),w, h);
    delete u;
    printf(" get_mask \n");

    return area;
}

XImage *ROIMedDrawingArea::get_XImage()
{
    int w = get_width();
    int h = get_height();
    XImage *ximage;
    Widget wid = baseWidget();
    if( (ximage = XGetImage(XtDisplay(wid), XtWindow(wid),
                           0, 0, w, h, AllPlanes, ZPixmap)) == NULL)
    {
        return NULL;
    }
    return ximage;
}

```

```

void ROIMedDrawingArea::expose(Widget w, XtPointer callData)
{
    MedDrawingArea::expose(w, callData);
    printf("    ROIMedDrawingArea expose\n");
    ShowROI();
}

void ROIMedDrawingArea::copyArea(int x, int y, int w, int h)
{
    int w1 = get_width();
    int h1 = get_height();
    if(x < 0) x = 0;
    else if(x > (w1-1)) x = w1-1;
    if(y < 0) y = 0;
    else if(y > (h1-1)) y = h1-1;
    if((x+w) < w1 && (y+h) < h1)
        DrawingArea::copyArea(x, y, w, h);
}

void ROIMedDrawingArea::display()
{
    ShowROI();
}

void ROIMedDrawingArea::input(Widget w, XtPointer callData)
{
    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;

    int xpos = cb->event->motion.x;
    int ypos = cb->event->motion.y;

    if (cb->event->type == ButtonPress)
    {
        if (cb->event->xbutton.button == Button3)
        {
            finished(xpos, ypos);
        }
        else if (cb->event->xbutton.button == Button2)
        {
            _button2Pressed = TRUE;
            midpressed(xpos, ypos);
        }
        else if (cb->event->xbutton.button == Button1)
        {
            _button1Pressed = TRUE;
            pressed(xpos, ypos);
        }
    }
    else if (cb->event->type == ButtonRelease)
    {
        if (cb->event->xbutton.button == Button3)
        {
            //printf(" R Button3\n");
        }
        else if (cb->event->xbutton.button == Button2)
        {
            //printf(" R Button2\n");
            _button2Pressed = FALSE;
        }
        else if (cb->event->xbutton.button == Button1)
        {
            _button1Pressed = FALSE;
            if(_ROI != NULL) released(xpos, ypos);
        }
    }
}

```

```

}

void ROIMedDrawingArea::motion ( Widget w, XEvent *event )
{
    //---- Start editable code block: DrawingArea.resize

    //XmDrawingAreaCallbackStruct *cbs = (XmDrawingAreaCallbackStruct*) callData;

    //--- Comment out the following line when DrawingArea::resize is implemented:

    //::VkUnimplemented ( w, "DrawingArea::resize" );

    int xpos = event->xmotion.x;
    int ypos = event->xmotion.y;

    if(_button1Pressed) moved(xpos, ypos);
    if(_button2Pressed) midmoved(xpos, ypos);

    //if(_objMag->_win3D != NULL) ((Bb3DLocalizer *) (_objMag->_localizer3d))->update_RC

    if(_objMag -> msgsRight.show_detail) show_info(xpos, ypos);

    //---- End editable code block: DrawingArea.resize
}

// End ROIMedDrawingArea::resize()

void ROIMedDrawingArea::show_info(int x, int y)
{
    int r, g, b;

    r=g=b=0;
    if(_cimg -> red != NULL) r = (_cimg -> red)[y][x];
    if(_cimg -> green != NULL) g = (_cimg -> green)[y][x];
    if(_cimg -> blue != NULL) b = (_cimg -> blue)[y][x];

    int signal = _zoomImg[y][x];

    ((BbDetail *) (_objMag->_LDet1)) -> set(x, y, r, g, b, signal);
}

void ROIMedDrawingArea::pressed(int xpos, int ypos)
{
    if(_ROI == NULL || (_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_REDEF
    {
        CreateROI(_roi_type);
        _ROI -> init(xpos, ypos);
    }
    else if(_ROI != NULL && !(_ROI -> _draw_status) && _roi_action == ROI_REDEFINE )
    {
        _ROI -> new_started(xpos, ypos);
    }
    else if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
    {
        //display();
        _ROI -> init_modify(xpos, ypos);
    }
}

void ROIMedDrawingArea::midpressed(int xpos, int ypos)
{
    if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
    {
        _ROI -> init_move(xpos, ypos);
    }
}

```

```

void ROIMedDrawingArea::released(int xpos, int ypos)
{
    if(_ROI != NULL && !(_ROI -> _draw_status))
        _ROI -> released(xpos, ypos);
    else if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
        _ROI -> released_modify(xpos, ypos);
}

void ROIMedDrawingArea::finished(int xpos, int ypos)
{
    //if(_ROI != NULL && !(_ROI -> _draw_status))
    if(_ROI != NULL)
        _ROI -> finished(xpos, ypos);
    //else if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
    //    display();
}

void ROIMedDrawingArea::moved(int xpos, int ypos)
{
    if(_ROI != NULL && !(_ROI -> _draw_status))
        _ROI -> motion(xpos, ypos);
    else if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
        _ROI -> motion_modify(xpos, ypos);
}

void ROIMedDrawingArea::midmoved(int xpos, int ypos)
{
    if(_ROI != NULL && _ROI -> _draw_status && _roi_action == ROI_MODIFY)
        _ROI -> motion_move(xpos, ypos);
}

void ROIMedDrawingArea::CreateROI (int roi_type)
{
    EraseROI();

    Widget w = baseWidget();

    switch (roi_type)
    {
        case ROI_RECTANGLE:
            _ROI = new Rectangle(w, _roi_color);
            break;
        case ROI_ELLIPSE:
            _ROI = new Ellipse(w, _roi_color);
            break;
        case ROI_POLYGON:
            _ROI = new Polygon(w, _roi_color);
            break;
        case ROI_FREEHAND:
            _ROI = new FreeHand(w, _roi_color);
            break;
        default:
            _ROI = NULL;
            break;
    }
    _ROI -> _event = ROI_REDEFINE;
    _ROI -> _width = get_width();
    _ROI -> _height = get_height();

    _ROI -> _roiView = this;
}

void ROIMedDrawingArea::CreateROI2(int roi_type)
{
    if(_ROI != NULL)

```

```
{
    delete _ROI;
    _ROI = NULL;
}
```

```
Widget w = baseWidget();
```

```
switch (roi_type)
{
    case ROI_RECTANGLE:
        _ROI = new Rectangle(w, _roi_color);
        break;
    case ROI_ELLIPSE:
        _ROI = new Ellipse(w, _roi_color);
        break;
    case ROI_POLYGON:
        _ROI = new Polygon(w, _roi_color);
        break;
    case ROI_FREEHAND:
        _ROI = new FreeHand(w, _roi_color);
        break;
    default:
        _ROI = NULL;
        break;
}
_ROI -> _event = ROI_REDEFINE;
_ROI -> _width = get_width();
_ROI -> _height = get_height();

_ROI -> _roiView = this;
}
```

```
void ROIMedDrawingArea::AcceptROI()
{
    if(_ROI != NULL)
    {
        if(_ROI -> _show_status)
        {
            if(_cimg2 != NULL)
            {
                delete _cimg2;
                _cimg2 = NULL;
            }

            if(!_ROI -> _draw_status)
                _ROI -> finished(0, 0);

            _ROI -> fill();

            if(_ROI -> _area != NULL)
            {
                Utility_Vision *u = new Utility_Vision();
                u -> freeImg(_ROI -> _area);
                delete u;
            }
            _ROI -> _area = get_mask();

            display();
        }
    }
}
```

```
void ROIMedDrawingArea::EraseROI()
{
    if(_ROI != NULL)
```

```
{
    delete _ROI;
    _ROI = NULL;
    DrawingArea::display();
}
```

```
void ROI MedDrawingArea::ShowROI()
{
    DrawingArea::display();

    if(_ROI != NULL)
    {
        if(_ROI -> _draw_status)
        {
            _ROI -> draw();
            _ROI -> _show_status = TRUE;
        }
    }
}
```

```
void ROI MedDrawingArea::HideROI()
{
    DrawingArea::display();
}
```

```
void ROI MedDrawingArea::set_color(int color)
{
    _roi_color = color;
    if(_ROI != NULL)
    {
        if(_ROI->_gc != NULL) XtReleaseGC(_ROI->_widget, _ROI->_gc);
        Utility_Widget *uw = new Utility_Widget();
        _ROI -> _gc = uw -> get_GC(_ROI->_widget, color);
        delete uw;
    }
}
```


User: meide
Host: phoenix
Class: phoenix
Job: DrawingAreaUI.C

```

////////////////////////////////////
// LineDrawingArea.c++
////////////////////////////////////

```

```
#include "LineDrawingArea.h"
```

```
#include "Utility_Math.h"
```

```
#include "Utility_Widget.h"
```

```
#include <stdio.h>
```

```
LineDrawingArea::LineDrawingArea(int w, int h, const char *name, Widget wid, int type,
: DrawingAreaUI(name, wid, flag)
```

```
{
    _draw_type = type;
    _width = w;
    _height = h;
    _x = NULL;
    _y = NULL;
    _drawX = NULL;
    _drawY = NULL;
}
```

```
LineDrawingArea::~~LineDrawingArea()
```

```
{
    if(_x != NULL) delete _x;
    if(_y != NULL) delete _y;
    if(_drawX != NULL) delete _drawX;
    if(_drawY != NULL) delete _drawY;
}
```

```
void LineDrawingArea::set(int sz, float *x, float *y)
```

```
{
    _size = sz;

    if(_x != NULL) delete _x;
    if(_y != NULL) delete _y;

    _x = x;
    _y = y;
    printf(" LineDrawingArea set 2\n");
}
```

```
Utility_Math *um = new Utility_Math();
```

```
if(x != NULL) um -> get_minmax(sz, x, &_minX, &_maxX);
if(y != NULL) um -> get_minmax(sz, y, &_minY, &_maxY);
```

```
float c1, c2, d1, d2;
```

```
um -> lineParaFromTwoPoints(0, 0, float(_size-1), float(_width-1), &c1, &c2);
um -> lineParaFromTwoPoints(_minY, 0, _maxY, float(_height-1), &d1, &d2);
```

```
if(_drawX != NULL) delete _drawX;
if(_drawY != NULL) delete _drawY;
```

```
printf(" LineDrawingArea set 3\n");
```

```
_drawX = new int[_size];
_drawY = new int[_size];
```

```
for(int i=0; i<_size; i++)
{
    _drawX[i] = um -> int_t( c1 * float(i) + c2);
    _drawY[i] = um -> int_t( d1 * _y[i] + d2);
}
```

```

    delete um;
}

void LineDrawingArea::expose(Widget w, XtPointer callData)
{
    display();
}

void LineDrawingArea::display(int x, int y)
{
    XtVaSetValues (baseWidget(),
                   XmNx, x,
                   XmNy, y,
                   XmNwidth, _width,
                   XmNheight, _height,
                   (XtPointer) NULL );

    //show();
}

void LineDrawingArea::display(int color)
{
    Utility_Widget *uw = new Utility_Widget();

    XFillRectangle(XtDisplay(baseWidget()), XtWindow(baseWidget()),
                   uw->get_GC(baseWidget(), COLOR_BLACK), 0, 0, _width, _height);

    if(_draw_type == DRAW_BAR) draw_bar(color);
    else if(_draw_type == DRAW_CURVE) draw_curve(color);

    delete uw;
}

void LineDrawingArea::draw_bar(int color)
{
    Utility_Widget *uw = new Utility_Widget();
    GC gc = uw -> get_GC(baseWidget(), color);

    for(int i=0; i<_size; i++)
        uw -> draw_line(baseWidget(), gc, _drawX[i], _height,
                        _drawX[i], _height - _drawY[i]);

    XtReleaseGC(baseWidget(), gc);
    delete uw;
}

void LineDrawingArea::draw_curve(int color)
{
    Utility_Widget *uw = new Utility_Widget();
    GC gc = uw -> get_GC(baseWidget(), color);

    for(int i=1; i<_size; i++)
        uw -> draw_line(baseWidget(), gc, _drawX[i-1], _height - _drawY[i-1],
                        _drawX[i], _height - _drawY[i]);

    XtReleaseGC(baseWidget(), gc);
    delete uw;
}

void LineDrawingArea::draw_onePoint(int i, GC gc)
{
    XDrawLine(XtDisplay(baseWidget()), XtWindow(baseWidget()), gc,
              _drawX[i-1], _height - _drawY[i-1], _drawX[i], _height - _drawY[i]);
}

void LineDrawingArea::resize(Widget w, XtPointer callData)
{

```

```

}

void LineDrawingArea::input(Widget w, XtPointer callData)
{
    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;

    int xpos = cb->event->motion.x;
    int ypos = cb->event->motion.y;

    if (cb->event->type == ButtonPress)
    {
        if (cb->event->button.button == Button3)
        {
        }
        else if (cb->event->button.button == Button2)
        {
        }
        else if (cb->event->button.button == Button1)
        {
            _button1Pressed = TRUE;
        }
    }
    else if (cb->event->type == ButtonRelease)
    {
        if (cb->event->button.button == Button3)
        {
        }
        else if (cb->event->button.button == Button2)
        {
        }
        else if (cb->event->button.button == Button1)
        {
            _button1Pressed = FALSE;
        }
    }
}

void LineDrawingArea::motion ( Widget w, XEvent *event )
{
    //---- Start editable code block: DrawingArea resize

    //--- Comment out the following line when DrawingArea::resize is implemented:

    int xpos = event->motion.x;
    int ypos = event->motion.y;

    if(_button1Pressed)
    {
    }
}

```

```

////////////////////////////////////
// TwoLinesLineDrawingArea.h
////////////////////////////////////

```

```
#include "TwoLinesLineDrawingArea.h"
```

```
#include "Utility_Math.h"
```

```
#include "Utility_Vision.h"
```

```
#include "Utility_Widget.h"
```

```
#include <stdio.h>
```

```
TwoLinesLineDrawingArea::TwoLinesLineDrawingArea(int w, int h, const char *name, Widget
: LineDrawingArea(w, h, name, wid, type, flag)
```

```
{
    _twolines = NULL;
}
```

```
TwoLinesLineDrawingArea::~TwoLinesLineDrawingArea()
```

```
{
    if(_twolines != NULL) delete _twolines;
}
```

```
void TwoLinesLineDrawingArea::newTwoLines(float low, float high, float minI, float maxI)
```

```
{
    if(_twolines != NULL) delete _twolines;
    _twolines = new TwoLines(baseWidget(), _width-1, _height-1);

    float c1, c2;

    Utility_Math *um = new Utility_Math();
    um -> lineParaFromTwoPoints(minI, 0, maxI, float(_width-1), &c1, &c2);
```

```
    float x1 = c1 * low + c2;
    float x2 = c1 * high + c2;
```

```
    printf("newTwoLines:: low high %f %f mm %f %f %d %d \n", low, high,
        minI, maxI, um->int_t(x1), um->int_t(x2) );
    _twolines -> set( um->int_t(x1), um->int_t(x2) );
```

```
    delete um;
}
```

```
void TwoLinesLineDrawingArea::expose(Widget w, XtPointer callData)
```

```
{
    LineDrawingArea::expose(w, callData);
    _twolines -> draw();
}
```

```
void TwoLinesLineDrawingArea::resize(Widget w, XtPointer callData)
```

```
{
}
```

```
void TwoLinesLineDrawingArea::input(Widget w, XtPointer callData)
```

```
{
    XmDrawingAreaCallbackStruct *cb = (XmDrawingAreaCallbackStruct*) callData;

    int xpos = cb->event->xmotion.x;
    int ypos = cb->event->xmotion.y;
}
```

```

if (cb->event->type == ButtonPress)
{
    if (cb->event->xbutton.button == Button3)
    {

    }
    else if (cb->event->xbutton.button == Button2)
    {

    }
    else if (cb->event->xbutton.button == Button1)
    {
        _button1Pressed = TRUE;
        _twolines->init(xpos);
    }
}
else if (cb->event->type == ButtonRelease)
{
    if (cb->event->xbutton.button == Button3)
    {

    }
    else if (cb->event->xbutton.button == Button2)
    {

    }
    else if (cb->event->xbutton.button == Button1)
    {
        _button1Pressed = FALSE;
    }
}
}

void TwoLinesLineDrawingArea::motion ( Widget w, XEvent *event )
{
    //---- Start editable code block: DrawingArea resize

    //--- Comment out the following line when DrawingArea::resize is implemented:

    int xpos = event->xmotion.x;
    int ypos = event->xmotion.y;

    if(_button1Pressed)
    {
        _twolines -> draw();
        _twolines -> draw(xpos);
    }
}

```

```

////////////////////////////////////
// HistoTwoLinesDrawingArea.c++
////////////////////////////////////

```

```
#include "HistoTwoLinesDrawingArea.h"
```

```
#include "ImgAlloc.h"
```

```
#include "Utility.h"
```

```
#include "Utility_Math.h"
```

```
#include "Utility_Vision.h"
```

```
#include "Utility_Widget.h"
```

```
#include <stdio.h>
```

```
HistoTwoLinesDrawingArea::HistoTwoLinesDrawingArea(int w, int h, const char *name, Width
: TwoLinesLineDrawingArea(w, h, name, width, type, flag)
```

```
{
    _mapImg = alloc_shimg(w, h);
    _map = NULL;
    _label_min = NULL;
    _label_max = NULL;
    _label_low = NULL;
    _label_high = NULL;
}
```

```
HistoTwoLinesDrawingArea::~HistoTwoLinesDrawingArea()
```

```
{
    if(_mapImg != NULL) free_shimg(_mapImg);
    if(_map != NULL) delete _map;
}
```

```
void HistoTwoLinesDrawingArea::set(int w, int h, short **img, int size,
    unsigned char **mask, float minI, float maxI)
```

```
{
    _w = h;
    _h = h;
    _img = img;

    printf(" %d %f %f \n", size, minI, maxI);

    float *y = get_histogram(w, h, img, size, minI, maxI, &_minI, &_maxI, mask);
    if(y == NULL) return;

    printf(" %f %f\n", _minI, _maxI);

    float *x = new float[size];

    float c1, c2;

    Utility_Math *um = new Utility_Math();
    um -> lineParaFromTwoPoints(0, _minI, float(size-1), _maxI, &c1, &c2);
    delete um;

    for(int i=0; i<size; i++)
        x[i] = c1 * float(i) + c2;

    LineDrawingArea::set(size, x, y);
}
```

```
void HistoTwoLinesDrawingArea::expose(Widget w, XtPointer callData)
```

```
{
    TwoLinesLineDrawingArea::expose(w, callData);
}
```

```
void HistoTwoLinesDrawingArea::newTwoLines(float center, float width)
```

```

{
    TwoLinesLineDrawingArea newTwoLines(center, width, _minI, _maxI);
    set_mm();
    set_lowhigh();
    if(_twolines != NULL) update_map();
}

void HistoTwoLinesDrawingArea::update_lowhigh(float low, float high)
{
    float c1, c2;

    Utility_Math *um = new Utility_Math();
    um -> lineParaFromTwoPoints(_minI, 0, _maxI, float(_width-1), &c1, &c2);

    float x1 = c1 * low + c2;
    float x2 = c1 * high + c2;

    _twolines -> draw();
    _twolines -> set( um->int_t(x1), um->int_t(x2) );
    _twolines -> draw();

    delete um;
}

void HistoTwoLinesDrawingArea::update_map()
{
    int w = _width;
    int h = 30;
    int i, j, tmp1, tmp2, tmp3;
    int x1 = _twolines -> _x1;
    int x2 = _twolines -> _x2;

    for(i=0; i<h; i++)
        for(j=0; j<w; j++)
            _mapImg[i][j] = j;

    if(_whoami == MY_LEFT)
    {
        tmp1 = _objMag -> msgsLeft.img_visual_type;
        tmp2 = _objMag -> msgsLeft.img_scale_type;
        tmp3 = 0;
    }
    else if(_whoami == MY_RIGHT)
    {
        tmp1 = _objMag -> msgsRight.img_visual_type;
        tmp2 = _objMag -> msgsRight.img_scale_type;
        tmp3 = _objMag -> msgsRight.flowDir;
    }

    if(_map == NULL)
    {
        if(_whoami == MY_LEFT)
            _map = new MedDrawingArea("GE", _objMag -> _LHist -> baseWidget(), 0);
        else if(_whoami == MY_RIGHT)
            _map = new MedDrawingArea("GE", _objMag -> _RHist -> baseWidget(), 0);

        _map -> set(w, h, _mapImg, tmp1, tmp2, 1.0, float(x1), float(x2), tmp3);

        _map -> show();
        ((DrawingArea *)_map) -> display(0, 120);
    }
    else
    {
        _map -> set(w, h, _mapImg, tmp1, tmp2, 1.0, float(x1), float(x2), tmp3);
        _map -> display();
    }
}

```



```

}
}

void HistoTwoLinesDrawingArea::set_mm()
{
    Utility_Math *um = new Utility_Math();
    Utility_Widget *uw = new Utility_Widget();

    if(_label_min != NULL)
        uw -> set_label(_label_min, um -> int_t(_minI));
    if(_label_max != NULL)
        uw -> set_label(_label_max, um -> int_t(_maxI));

    delete um;
    delete uw;
}

void HistoTwoLinesDrawingArea::set_lowhigh()
{
    Utility_Math *um = new Utility_Math();
    Utility_Widget *uw = new Utility_Widget();

    int x1 = _twolines -> _x1;
    int x2 = _twolines -> _x2;

    float c1, c2;
    um -> lineParaFromTwoPoints(0, _minI, float(_width-1), _maxI, &c1, &c2);

    float low = c1 * float(x1) + c2;
    float high = c1 * float(x2) + c2;

    //printf(" LOW  HIGH    %f    %f\n", low, high);

    if(_label_low != NULL)
        uw -> set_label(_label_low, um -> int_t(low));
    if(_label_high != NULL)
        uw -> set_label(_label_high, um -> int_t(high));

    delete um;
    delete uw;
}

void HistoTwoLinesDrawingArea::resize(Widget w, XtPointer callData)
{
}

void HistoTwoLinesDrawingArea::input(Widget w, XtPointer callData)
{
    TwoLinesLineDrawingArea::input(w, callData);
}

void HistoTwoLinesDrawingArea::motion ( Widget w, XEvent *event )
{
    TwoLinesLineDrawingArea::motion(w, event);
    if(_button1Pressed) change();
}

void HistoTwoLinesDrawingArea::change()
{
    if(_twolines != NULL)
    {
        set_lowhigh();
        update_map();
        update_imgView();
    }
}

```

```

}

void HistoTwoLinesDrawingArea::update_imgView()
{
    float c1, c2;

    Utility_Math *um = new Utility_Math();
    um -> lineParaFromTwoPoints(0, _minI, float(_width-1), _maxI, &c1, &c2);
    delete um;

    int x1 = _twolines -> _x1;
    int x2 = _twolines -> _x2;

    float low = c1 * float(x1) + c2;
    float high = c1 * float(x2) + c2;

    if(_whoami == MY_LEFT && _objMag -> msgsLeft.img_space == IMAGE_2D)
    {
        _objMag -> msgsLeft.img_winCenter = low;
        _objMag -> msgsLeft.img_winWidth = high;
        _objMag -> update_Llowhigh();
        _objMag -> _imgView -> update(float(low), float(high));
    }
    else if(_whoami == MY_RIGHT && _objMag -> msgsRight.img_space == IMAGE_2D)
    {
        _objMag -> msgsRight.img_winCenter = low;
        _objMag -> msgsRight.img_winWidth = high;
        _objMag -> update_Rlowhigh();
        _objMag -> _imgView2 -> update(float(low), float(high));
    }
}

float *HistoTwoLinesDrawingArea::get_histogram(int w, int h, short **img, int size,
    float min_img, float max_img, float *minI, float *maxI, unsigned char **mask)
{
    float minI0, maxI0, val;
    int i, j, s;

    Utility_Vision *uv = new Utility_Vision();
    uv -> get_bound(w, h, img, &minI0, &maxI0);
    delete uv;

    if(min_img != max_img)
    {
        if(minI0 < min_img) minI0 = min_img;
        if(maxI0 > max_img) maxI0 = max_img;
    }

    s = (int)(maxI0 - minI0);
    if(s < 2) return NULL;

    float *histo = new float[size];

    for(i=0; i<size; i++)
        histo[i] = 0;

    float c1, c2;

    Utility_Math *um = new Utility_Math();
    um -> lineParaFromTwoPoints(minI0, 0, maxI0, size-1, &c1, &c2);

    for(i=0; i<h; i++)
    for(j=0; j<w; j++)
    {
        if(mask == NULL)

```

```
{
    val = (float)img[i][j];
    s = um -> int_t( c1 * val + c2);
    if( s >= 0 && s <= (size-1) )
        ++histo[s];
}
else if(mask[i][j] == 1)
{
    val = (float)img[i][j];
    s = um -> int_t( c1 * val + c2);
    if( s >= 0 && s <= (size-1) )
        ++histo[s];
}
}

delete um;

*minI = minI0;
*maxI = maxI0;

return histo;
}
```

```
#include "Utility_Math.h"
```

```
#include <math.h>
```

```
Utility_Math::Utility_Math()
```

```
{
}
```

```
Utility_Math::~~Utility_Math()
```

```
{
}
```

```
int Utility_Math::int_t(float x)
```

```
{
    if(x >= 0) return (int)(x+0.5);
    else return (int)(x-0.5);
}
```

```
void Utility_Math::get_minmax(int sz, float *x, float *minX, float *maxX)
```

```
{
    *minX = x[0];
    *maxX = x[0];
    for(int i=1; i<sz; i++)
    {
        if(*minX > x[i]) *minX = x[i];
        if(*maxX < x[i]) *maxX = x[i];
    }
}
```

```
int Utility_Math::solve_poly2(float a, float b, float c,
    float *x1, float *x2)
```

```
{
    //
    // Problem:
    // Givein:  $a * x^2 + b * x + c = 0$ 
    // Find:  $x$ 
    //

```

```
float tmp = b*b - 4*a*c;
if(tmp < 0) return 0;
else if(a == 0)
{
    if(b == 0) return 0;
    else {*x1 = *x2 = -c/b;}
}
```

```
else
{
    *x1 = (-b + fsqrt(tmp))/2/a;
    *x2 = (-b - fsqrt(tmp))/2/a;
}
```

```
return 1;
```

```
}
```

```
int Utility_Math::lineParaFromTwoPoints(float x1, float y1, float x2, float y2,
    float *c1, float *c2)
```

```
{
    if(fabsf(x2-x1) <= 1.e-10)
    {
        *c1 = x1;
        return 0;
    }
    *c1 = (y2-y1)/(x2-x1);
    *c2 = y1 - (*c1) * x1;
    return 1;
}
```

```
}  
  
int Utility_Math::lineParamFromTwoPoints(float x1, float y1, float sita,  
float *c1, float *c2)  
{  
    float pi = 3.141592654;  
  
    if(fabsf(sita - pi/2.0) <= 1.e-5)  
    {  
        *c1 = x1;  
        return 0;  
    }  
    *c1 = tanf(sita);  
    *c2 = y1 - (*c1) * x1;  
    return 1;  
}
```

```
float Utility_Math::get_angle(float x1, float y1, float x2, float y2)  
{  
    float dy = y2 - y1;  
    float dx = x2 - x1;  
    float alpha = atan2f(dy, dx);  
    float pi = 3.141592654;  
  
    if(alpha < 0) alpha += 2*pi;  
    return alpha;  
}
```

User: meide
Host: phoenix
Class: phoenix
Job: LineDrawingArea.C

```

#include "Utility_Vision.h"

#include "ImgAlloc.h"
#include "Utility_Math.h"
#include <stdio.h>

Utility_Vision::Utility_Vision()
{
}

Utility_Vision::~Utility_Vision()
{
}

void Utility_Vision::freeImg(unsigned char **grayimg)
{
    free_img(grayimg);
}

void Utility_Vision::freeCImg(ColorImage *img)
{
    if(img -> red != NULL) free_img(img -> red);
    if(img -> green != NULL) free_img(img -> green);
    if(img -> blue != NULL) free_img(img -> blue);
    if(img != NULL) delete img;
}

void Utility_Vision::freeShimg(short **img)
{
    free_shimg(img);
}

void Utility_Vision::get_bound(int w, int h, short **img, float *min_I, float *max_I)
{
    *min_I = 1.0e30;
    *max_I = -1.0e30;
    int k;
    short *pf;

    for(k=0, pf=*img; k<(w*h); k++, pf++) {
        if((float)(*pf) < *min_I) {*min_I = *pf;}
        if((float)(*pf) > *max_I) {*max_I = *pf;}
    }
}

ColorImage *Utility_Vision::toVisual(int visual_method, int w, int h, short **shimg, fl
{
    if(visual_method == VISUAL_GRAY)
    {
        ColorImage *img = new ColorImage;

        img -> red = toGray(w, h, shimg, p1, p2);
        if(img -> red == NULL) {delete img; return NULL;}

        img -> green = img -> red;
        img -> blue = img -> red;

        return img;
    }
    else if(visual_method == VISUAL_COLOR)
    {
        return toColor(w, h, shimg, p1, p2);
    }

    return NULL;
}

```

```

}

unsigned char **Utility_Vision::toGray(int w, int h, short **shimg, float widCenter, fl
{
    unsigned char    **grayimg;
    float            val, tmp;
    int              i, j;
    float            min_sig, max_sig;

    if(widCenter == widWidth && widWidth == 0)
    {
        get_bound(w,h,shimg,&min_sig,&max_sig);
    }
    else
    {
        min_sig = widCenter;
        max_sig = widWidth;
    }

    if(fabsf(max_sig - min_sig) < 1.e-10) return NULL;

    grayimg = alloc_img(w, h);
    if(grayimg == NULL) return NULL;

    for (i=0; i<h; i++)
    for(j=0; j<w; j++)
    {
        val = shimg[i][j];
        if (val <= min_sig) tmp = 0;
        else if(val >= max_sig) tmp = 255.0;
        else
            tmp = (val - min_sig)/(max_sig - min_sig) * 255.0;

        grayimg[i][j] = tmp;
    }

    return grayimg;
}

ColorImage *Utility_Vision::toColor(int w, int h, short **fimg, float in_low, float ir
{
    float            tmp, low, high;
    short            val;
    unsigned char    r,g,b;
    int              i, j;

    if(in_low == in_high && in_high == 0)
    {
        get_bound(w,h,fimg, &low, &high);
    }
    else
    {
        low = in_low;
        high = in_high;
    }

    int n = 6;
    int *thresh = new int[n+1];

    float    intensity0 = 155;

    float dx = (high - low)/(float)n;

    float scale = ((float)(250-intensity0))/dx;

```



```

for(i=0; i<(n+1); i++)
    thresh[i] = (int)((float)i*dx);

```

50

```

ColorImage *img = new ColorImage;

```

```

img -> red = alloc_img(w, h);
img -> green = alloc_img(w, h);
img -> blue = alloc_img(w, h);

```

```

if(img -> red == NULL || img -> green == NULL || img -> blue == NULL)
{
    delete img;
    delete thresh;
    return NULL;
}

```

```

for(i=0; i<h; i++)
for(j=0; j<w; j++) {

```

```

    val = fimg[i][j];

```

```

    if(val >= thresh[n])
    {

```

```

        tmp = ((float)val - (float)thresh[n])/2.0 + 160;
        if(tmp >= 255) tmp = 250;

```

```

        r = (unsigned char)tmp;
        g = (unsigned char)tmp;
        b = (unsigned char)tmp;
    }

```

```

    else if(val >= thresh[n-1] && val < thresh[n])
    {

```

```

        tmp = ((float)val - (float)thresh[n-1])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = 0;
        b = 0;
    }

```

```

    else if(val >= thresh[n-2] && val < thresh[n-1])
    {

```

```

        tmp = ((float)val - (float)thresh[n-2])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = 0;
        b = (unsigned char)tmp;
    }

```

```

    else if(val >= thresh[n-3] && val < thresh[n-2])
    {

```

```

        tmp = ((float)val - (float)thresh[n-3])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = (unsigned char)tmp;
        b = 0;
    }

```

```

    else if(val >= thresh[n-4] && val < thresh[n-3])
    {

```

```

        tmp = ((float)val - (float)thresh[n-4])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = 0;
        g = (unsigned char)tmp;
        b = 0;
    }

```

```

    else if(val >= thresh[n-5] && val < thresh[n-4])
    {

```

```

        tmp = ((float)val - (float)thresh[n-5])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
    }
}

```

```

    r = 0;
    g = (unsigned char)tmp;
    b = (unsigned char)tmp;
}
else if(val >= thresh[n-6] && val < thresh[n-5])
{
    r = 0;
    g = 0;
    tmp = ( (float)val - (float)thresh[n-6] ) * scale + intensity0;
    if(tmp >= 255) tmp = 255;
    b = (unsigned char)tmp;
}
else if(val < thresh[n-6])
{
    tmp = ((float)val - (float)thresh[n-6]) / 2.0 + 100;
    if(tmp >= 255) tmp = 255;
    if(tmp < 0) tmp = 0;

    r = (unsigned char)tmp;
    g = (unsigned char)tmp;
    b = (unsigned char)tmp;
}
else
{
    r=g=b=0;
    printf("Color STRANEG\n");
}

img -> red[i][j] = r;
img -> green[i][j] = g;
img -> blue[i][j] = b;
}

delete thresh;
return img;
}

ColorImage *Utility_Vision::toColor2(int w, int h, short **fimg, float in_low, float i
{
    float        tmp, low, high;
    short        val;
    unsigned char r,g,b;
    int          i, j;

    if(in_low == in_high && in_high == 0)
    {
        get_bound(w,h,fimg, &high, &low);
        low = -low;
        high = -high;
    }
    else
    {
        low = -in_high;
        high = -in_low;
    }

    int n = 6;
    int *thresh = new int[n+1];

    float intensity0 = 155;

    float dx = (high - low) / (float)n;

    float scale = ((float)(250-intensity0)) / dx;

```

```

for(i=0; i<(n+1); i++)
    thresh[i] = (int)(LOW + (float)i*dx);

ColorImage *img = new ColorImage;

img -> red = alloc_img(w, h);
img -> green = alloc_img(w, h);
img -> blue = alloc_img(w, h);

if(img -> red == NULL || img -> green == NULL || img -> blue == NULL)
{
    delete img;
    delete thresh;
    return NULL;
}

for(i=0; i<h; i++)
for(j=0; j<w; j++) {

    val = -fimg[i][j];

    if(val >= thresh[n])
    {
        tmp = ((float)val - (float)thresh[n])/2.0 + 160;
        if(tmp >= 255) tmp = 250;

        r = (unsigned char)tmp;
        g = (unsigned char)tmp;
        b = (unsigned char)tmp;
    }
    else if(val >= thresh[n-1] && val < thresh[n])
    {
        tmp = ((float)val - (float)thresh[n-1])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = 0;
        b = 0;
    }
    else if(val >= thresh[n-2] && val < thresh[n-1])
    {
        tmp = ((float)val - (float)thresh[n-2])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = 0;
        b = (unsigned char)tmp;
    }
    else if(val >= thresh[n-3] && val < thresh[n-2])
    {
        tmp = ((float)val - (float)thresh[n-3])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = (unsigned char)tmp;
        g = (unsigned char)tmp;
        b = 0;
    }
    else if(val >= thresh[n-4] && val < thresh[n-3])
    {
        tmp = ((float)val - (float)thresh[n-4])*scale + intensity0;
        if(tmp >= 255) tmp = 250;
        r = 0;
        g = (unsigned char)tmp;
        b = 0;
    }
    else if(val >= thresh[n-5] && val < thresh[n-4])
    {
        tmp = ((float)val - (float)thresh[n-5])*scale + intensity0;

```

```

    if(tmp >= 255) tmp = 250;
    r = 0;
    g = (unsigned char)tmp;
    b = (unsigned char)tmp;
}
else if(val >= thresh[n-6] && val < thresh[n-5])
{
    r = 0;
    g = 0;
    tmp = ( (float)val - (float)thresh[n-6] ) * scale + intensity0;
    if(tmp >= 255) tmp = 250;
    b = (unsigned char)tmp;
}
else if(val < thresh[n-6])
{
    tmp = ((float)val - (float)thresh[n-6]) / 2.0 + 100;
    if(tmp >= 255) tmp = 250;
    if(tmp < 0) tmp = 0;

    r = (unsigned char)tmp;
    g = (unsigned char)tmp;
    b = (unsigned char)tmp;
}
else
{
    r=g=b=0;
    printf("Color STRANEG\n");
}

img -> red[i][j] = r;
img -> green[i][j] = g;
img -> blue[i][j] = b;
}

delete thresh;
return img;
}

short **Utility_Vision::scale_img(int flag, int w1, int h1, short **img1,
float zoom, int *w2, int *h2)
{
    Utility_Math *u = new Utility_Math();
    *w2 = u -> int_t(zoom * (float)w1);
    *h2 = u -> int_t(zoom * (float)h1);
    delete u;

    if(zoom > 1.0 && flag == SCALE_SPLINE)
        return stretching_img(w1,h1,img1,*w2,*h2);
    if(zoom > 1.0 && flag == SCALE_SIMPLE)
        return simple_stretching(w1, h1, img1, w2, h2);
    else if(zoom == 1.0) return copy_img(w1, h1, img1);
    else return shrinking_img(w1,h1,img1,*w2,*h2);
}

short **Utility_Vision::copy_img(int w, int h, short **img)
{
    short **img2 = (short **)alloc_shimg(w, h);
    if(img2 == NULL) return NULL;
    for(int i=0; i<h; i++)
        for(int j=0; j<w; j++)
            img2[i][j] = img[i][j];
    return img2;
}

```

```
short **Utility_Vision::shrinking_img(int w1,int h1,short **in_img,int w2, int h2)
{
    54
```

```
    int    i,j, i1, j1;
    float   rw,rh;
    short   **img;

    img = (short **)alloc_shimg(w2,h2);
    if(img == NULL) return NULL;

    rw = (float)w1/(float)w2;
    rh = (float)h1/(float)h2;

    Utility_Math *u = new Utility_Math();

    for(i=0; i<h2; i++)
    for(j=0; j<w2; j++)
    {
        i1 = u -> int_t(rh * (float)i);
        j1 = u -> int_t(rw * (float)j);
        img[i][j] = in_img[i1][j1];
    }

    delete u;
    return img;
}
```

```
inline short Utility_Vision::bilinear(float dx,float dy,int x1,int y1,short **img)
{
    float tmp,z11,z12,z21,z22;

    z11 = (float)img[x1][y1];
    z12 = (float)img[x1][y1+1];
    z21 = (float)img[x1+1][y1];
    z22 = (float)img[x1+1][y1+1];

    tmp = z11 + dx * (z21 - z11) + dy * (z12 - z11) +
          dx * dy * (z11 - z12 - z21 + z22);

    return (short)tmp;
}
```

```
short **Utility_Vision::stretching_img(int w1,int h1,short **in_img,int w2, int h2)
{
    /*
    int    i,j,i1,j1;
    float   rw,rh,fi1,fj1,di,dj;
    short   **img;
    Utility_Math *u = new Utility_Math();

    img = (short **)alloc_shimg(w2,h2);
    if(img == NULL) return NULL;

    rw = (float)w1/(float)w2;
    rh = (float)h1/(float)h2;

    for(i=0; i<h2; i++)
    for(j=0; j<w2; j++)
    {
        fi1 = (float)i * rh;
        fj1 = (float)j * rw;

        //i1 = u -> int_t(fi1);
        //j1 = u -> int_t(fj1);
        i1 = (int)fi1;
```

```

    j1 = (int)fj1;

    //if( i1 >= (h1-1) ) i1 = h1 - 2;
    //if( j1 >= (w1-1) ) j1 = w1 - 2;
    if( i1 == (h1-1) ) --i1;
    if( j1 == (w1-1) ) --j1;

    di = fi1 - (float)i1;
    dj = fj1 - (float)j1;

    img[i][j] = bilinear(di,dj,i1,j1,in_img);
}

delete u;
return img;
*/

short **img = (short **)alloc_shimg(w2,h2);
if(img == NULL) return NULL;

int i,j,k,l;
int zoom = int(float(w2)/float(w1));
int i0 = int(float(zoom)/2.0);
float dx, dy;

for(i=0; i<(h1-1); i++)
for(j=0; j<(w1-1); j++)
{
    for(k=0; k<zoom; k++)
    for(l=0; l<zoom; l++)
    {
        //img[i0+i*zoom+k][i0+j*zoom+l] = bilinear(float(k)/float(zoom),float(l)/float(z
        dx = float(k)/float(zoom);
        dy = float(l)/float(zoom);
        img[i0+i*zoom+k][i0+j*zoom+l] = float(in_img[i][j]) + dx * float(in_img[i+1][j])
        dy * float(in_img[i][j+1] - in_img[i][j]) + dx * dy * float(in_img[i][j] -
        in_img[i+1][j] + in_img[i+1][j+1]);
    }
}

for(i=0; i<i0; i++)
for(j=0; j<w1; j++)
    for(l=0; l<zoom; l++)
        img[i][j*zoom+l] = img[i0][j*zoom+l];

for(i=h2-1; i>=h2-1-i0; i--)
for(j=0; j<w1; j++)
    for(l=0; l<zoom; l++)
        img[i][j*zoom+l] = img[i0+(h1-2)*zoom+zoom-1][j*zoom+l];

for(i=0; i<h1; i++)
for(k=0; k<zoom; k++)
    for(j=0; j<i0; j++)
        img[i*zoom+k][j] = img[i*zoom+k][i0];

for(i=0; i<h1; i++)
for(k=0; k<zoom; k++)
    for(j=w2-1; j>=w2-1-i0; j--)
        img[i*zoom+k][j] = img[i*zoom+k][i0+(w1-2)*zoom+zoom-1];

return img;
}

short **Utility_Vision::simple_stretching(int w1,int h1,short **img1,int *w2, int *h2)

```

```

{
    Utility_Math *u = new Utility_Math();
    int zoom = u->int_t(float(*w2)/float(w1));
    delete u;

    *w2 = zoom * w1;
    *h2 = zoom * h1;

    short **img2 = (short **)alloc_shimg(*w2,*h2);
    if(img2 == NULL) return NULL;

    int i,j,k,l;

    for(i=0; i<h1; i++)
    for(j=0; j<w1; j++)
    {
        for(k=0; k<zoom; k++)
        for(l=0; l<zoom; l++)
            img2[i*zoom+k][j*zoom+l] = img1[i][j];
    }

    return img2;
}

void Utility_Vision::get_ROI(short **img, int x, int y, int w, int h, short **imgdata)
{
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
    {
        imgdata[i][j] = img[y+i][x+j];
    }
}

short **Utility_Vision::get_ROI(short **img, int x, int y, int w, int h, unsigned char
{
    short **imgdata = alloc_shimg(w, h);
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
    if(mask != NULL)
    {
        if(mask[i][j] == 1) imgdata[i][j] = img[y+i][x+j];
        else imgdata[i][j] = 0;
    }
    else
    {
        imgdata[i][j] = img[y+i][x+j];
    }
    return imgdata;
}

void Utility_Vision::highlight(float percent, int r1, int g1, int b1,
    int *r2, int *g2, int *b2 )
{
    float r, g, b;
    r = (255.0 - ((255.0 - (float)r1) * (100.0 - percent) / 100.0));
    g = (255.0 - ((255.0 - (float)g1) * (100.0 - percent) / 100.0));
    b = (255.0 - ((255.0 - (float)b1) * (100.0 - percent) / 100.0));
    if(r < 0) r = 0; else if(r > 255) r = 255;
    if(g < 0) g = 0; else if(g > 255) g = 255;
    if(b < 0) b = 0; else if(b > 255) b = 255;
    *r2 = r;
    *g2 = g;
    *b2 = b;
}

```

```
#include "Utility_Widget.h"
```

```
#include <Vk/VkFormat.h>
```

```
#include <Xm/TextF.h>
```

```
#include <stdio.h>
```

```
#include "Utility_Math.h"
```

```
#include "ImgAlloc.h"
```

```
Utility_Widget::Utility_Widget()
```

```
{
}
```

```
Utility_Widget::~~Utility_Widget()
```

```
{
}
```

```
GC Utility_Widget::get_xorGC(Widget w)
```

```
{
    XGCValues values;
    unsigned long    bgpix;

    bgpix = BlackPixel(XtDisplay(w), DefaultScreen(XtDisplay(w)));

    unsigned long a = 255;
    values.foreground = ( ((a<<8)<<8) + (a<<8) + a ) ^ bgpix;

    values.background = 0;
    values.function = GXxor;

    return XtGetGC(w, GCBackground | GCForeground | GCFunction, &values);
}
```

```
GC Utility_Widget::get_GC(Widget w, unsigned char r, unsigned char g, unsigned char b)
```

```
{
    XGCValues values;

    unsigned long a = ((unsigned long)r) + ((unsigned long)g)<<8 + ((unsigned long)b)<<16;

    values.foreground = a;
    //values.background = 0;

    values.function = GXcopy;
    return XtGetGC(w, GCForeground | GCFunction, &values);
}
```

```
GC Utility_Widget::get_GC(Widget w, int mode)
```

```
{
    XGCValues values;
    XColor exact, color;

    switch (mode)
    {
        case COLOR_RED:
            XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
                DefaultScreen(XtDisplay(w))), "red", &exact, &color);
            break;
        case COLOR_GREEN:
            XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
                DefaultScreen(XtDisplay(w))), "green", &exact, &color);
            break;
        case COLOR_BLUE:
            XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
                DefaultScreen(XtDisplay(w))), "blue", &exact, &color);
            break;
        case COLOR_BLACK:

```



```

        XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
        DefaultScreen(XtDisplay(w))), "black", &exact, &color);
        break;
    case COLOR_WHITE:
        XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
        DefaultScreen(XtDisplay(w))), "white", &exact, &color);
        break;
    case COLOR_YELLOW:
        XAllocNamedColor(XtDisplay(w), DefaultColormap(XtDisplay(w),
        DefaultScreen(XtDisplay(w))), "yellow", &exact, &color);
        break;
    default:
        break;
}

values.foreground = color.pixel;
values.background = 0;

values.function = GXcopy;
return XtGetGC(w, GCBackground | GCForeground | GCFunction, &values);
}

void Utility_Widget::set_label(Widget label, int i)
{
    XmString xms;
    char str[100];

    sprintf(str, "%d", i);
    xms=XmStringCreateSimple(str);
    XtVaSetValues (label,
                    XmNlabelString, xms,
                    (XtPointer) NULL );
}

void Utility_Widget::set_label(Widget label, float f)
{
    XmString xms;
    char str[100];

    sprintf(str, "%5.2f", f);
    xms=XmStringCreateSimple(str);
    XtVaSetValues (label,
                    XmNlabelString, xms,
                    (XtPointer) NULL );
}

void Utility_Widget::set_label(Widget label, char *str)
{
    XmString xms;

    xms=XmStringCreateSimple(str);
    XtVaSetValues (label,
                    XmNlabelString, xms,
                    (XtPointer) NULL );
}

void Utility_Widget::set_textfield(Widget textfield, int i)
{
    XmTextFieldSetString(textfield,
        (char *)VkFormat("%d", i));
}

void Utility_Widget::set_textfield(Widget textfield, float f)
{
    XmTextFieldSetString(textfield,

```

```

}

void Utility_Widget::draw_point(Widget w, GC gc, float x, float y)
{
    Utility_Math *u = new Utility_Math();
    XDrawPoint(XtDisplay(w), XtWindow(w), gc, u->int_t(x), u->int_t(y));
    delete u;
}

void Utility_Widget::draw_line(Widget w, GC gc, float x1, float y1,
    float x2, float y2)
{
    Utility_Math *u = new Utility_Math();
    XDrawLine(XtDisplay(w), XtWindow(w), gc, u->int_t(x1), u->int_t(y1),
        u->int_t(x2), u->int_t(y2));
    delete u;
}

void Utility_Widget::draw_rectangle(Widget wid, GC gc, float x, float y,
    float w, float h)
{
    Utility_Math *u = new Utility_Math();
    XDrawRectangle(XtDisplay(wid), XtWindow(wid), gc, u->int_t(x), u->int_t(y),
        u->int_t(w), u->int_t(h));
    delete u;
}

void Utility_Widget::draw_point(Widget w, float x, float y, int w1, int h1, ColorIn
{
    int x1 = int(x);
    int y1 = int(y);

    if(cimg != NULL && x1 >= 0 && x1 < w1 && y1 >= 0 && y1 < h1)
    {
        GC gc;

        printf(" <%d %d> :   %d %d %d\n", x1, y1, cimg->red[y1][x1], cimg->green[y1][x1],

        if(cimg->red != NULL && cimg->green != NULL && cimg->blue != NULL)
            gc = get_GC(w, cimg->red[y1][x1], cimg->green[y1][x1], cimg->blue[y1][x1]);
        else if(cimg->red != NULL && cimg->green == NULL && cimg->blue == NULL)
            gc = get_GC(w, cimg->red[y1][x1]);
        XDrawPoint(XtDisplay(w), XtWindow(w), gc, x1, y1);
        XtReleaseGC(w, gc);
    }
}

unsigned char **Utility_Widget::get_mask(Widget wid, int w, int h)
{
    XImage *ximage;

    //printf(" Utility_Widget::get_mask 1\n");
    if(w > 512) w = 512;
    if(h > 512) h = 512;

    if( (ximage = XGetImage(XtDisplay(wid), XtWindow(wid),
        0, 0, w, h, AllPlanes, ZPixmap)) == NULL)
    {
        return NULL;
    }
    //printf(" Utility_Widget::get_mask 2\n");
    Utility_Math *u = new Utility_Math();

```

```
unsigned char **area_img = (unsigned char **)alloc_img(w, h);
```

```
if(area_img == NULL)
```

```
{  
    delete u;  
    XDestroyImage(ximage);  
    return NULL;  
}
```

```
//printf(" Utility_Widget::get_mask 3\n");
```

```
int          i, j, i1, j1;  
unsigned long tmp;
```

```
for(i=0; i<h; i++)  
for(j=0; j<w; j++)  
{  
    tmp = XGetPixel(ximage, j, i);  
    if(tmp == 255) area_img[i][j] = 1;  
    else area_img[i][j] = 0;  
}
```

```
delete u;  
XDestroyImage(ximage);
```

```
return area_img;
```

```
}
```

```

#include "Utility.h"

#include <Vk/VkFormat.h>
#include <Xm/TextF.h>
#include "Utility_Math.h"

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>

Utility::Utility()
{
}

Utility::~Utility()
{
}

int Utility::get_ImgType(char *type)
{
    if(strcmp(type, "CT") == 0)
        return IMAGE_CT;
    else if(strcmp(type, "MR") == 0)
        return IMAGE_MR;
    else if(strcmp(type, "PCMRA") == 0)
        return IMAGE_PCMRA;
    else return IMAGE_NONE;
}

int Utility::get_ImgAnatomy(int img_type)
{
    switch (img_type)
    {
        case IMAGE_CT:
            return IMAGE_CT_HEAD;
        case IMAGE_MR:
            return IMAGE_MR_HEAD;
        case IMAGE_PCMRA:
            return IMAGE_PCMRA_HEAD;
        default:
            return IMAGE_NONE_NONE;
    }
}

int Utility::get_ImgAnatomy(int img_type, char *anatomy)
{
    switch (img_type)
    {
        case IMAGE_CT:
            if(strcmp(anatomy, "HEAD") == 0)
                return IMAGE_CT_HEAD;
            else if(strcmp(anatomy, "LUNG") == 0)
                return IMAGE_CT_LUNG;
            else return IMAGE_CT_NONE;
        case IMAGE_MR:
            if(strcmp(anatomy, "HEAD") == 0)
                return IMAGE_MR_HEAD;
            else if(strcmp(anatomy, "LUNG") == 0)
                return IMAGE_MR_LUNG;
            else return IMAGE_MR_NONE;
        case IMAGE_PCMRA:
            if(strcmp(anatomy, "HEAD") == 0)
                return IMAGE_PCMRA_HEAD;
            else if(strcmp(anatomy, "LUNG") == 0)

```

```

        return IMAGE_PCMRA_LUNG;
    else return IMAGE_PCMRA_NONE;
default:
    return IMAGE_NONE_NONE;
}
}

```

```

void Utility::get_GE(int img_type, int img_anatomy,
    float *widCenter, float *winWidth)
{

```

```

    printf("Utility:: %d %d\n",img_type, img_anatomy);

```

```

    switch (img_type)
    {

```

```

        case IMAGE_CT:

```

```

            switch (img_anatomy)
            {

```

```

                case IMAGE_CT_HEAD:

```

```

                    *widCenter = 1050;

```

```

                    *winWidth = 50;

```

```

                    break;

```

```

                case IMAGE_CT_LUNG:

```

```

                    *widCenter = 1050;

```

```

                    *winWidth = 50;

```

```

                    break;

```

```

                default:

```

```

                    *widCenter = 1050;

```

```

                    *winWidth = 50;

```

```

                    break;

```

```

            }

```

```

            break;

```

```

        case IMAGE_MR:

```

```

            switch (img_anatomy)
            {

```

```

                case IMAGE_MR_HEAD:

```

```

                    *widCenter = 1000;

```

```

                    *winWidth = 2000;

```

```

                    break;

```

```

                case IMAGE_MR_LUNG:

```

```

                    *widCenter = 1000;

```

```

                    *winWidth = 2000;

```

```

                    break;

```

```

                default:

```

```

                    *widCenter = 1000;

```

```

                    *winWidth = 2000;

```

```

                    break;

```

```

            }

```

```

            break;

```

```

        case IMAGE_PCMRA:

```

```

            switch (img_anatomy)
            {

```

```

                case IMAGE_PCMRA_HEAD:

```

```

                    *widCenter = 0;

```

```

                    *winWidth = 600;

```

```

                    break;

```

```

                case IMAGE_PCMRA_LUNG:

```

```

                    *widCenter = 0;

```

```

                    *winWidth = 1000;

```

```

                    break;

```

```

                default:

```

```

                    *widCenter = 0;

```

```

                    *winWidth = 1000;

```

```

                    break;

```

```

            }

```

```

            break;

```

```

        default:

```

```

*widCenter = 1000;
*winWidth = 2000;
break;
}
}

```

```

GE_PCMRA_HEADER_OBJ *Utility::copy_pc(GE_PCMRA_HEADER_OBJ *pc)
{

```

```

    GE_PCMRA_HEADER_OBJ *pc2 = new GE_PCMRA_HEADER_OBJ;

    pc2 -> img_width = pc -> img_width;
    pc2 -> img_height = pc -> img_height;
    pc2 -> slthick = pc -> slthick;
    pc2 -> pixsize_X = pc -> pixsize_X;
    pc2 -> pixsize_Y = pc -> pixsize_Y;
    pc2 -> heart_rate = pc -> heart_rate;
    pc2 -> pc_venc = pc -> pc_venc;
    pc2 -> min_I = pc -> min_I;
    pc2 -> max_I = pc -> max_I;
    pc2 -> mag_weighting_flag = pc -> mag_weighting_flag;
    pc2 -> venc_weighted_scale = pc -> venc_weighted_scale;
    pc2 -> ctr_R = pc -> ctr_R;
    pc2 -> ctr_A = pc -> ctr_A;
    pc2 -> ctr_S = pc -> ctr_S;
    pc2 -> norm_R = pc -> norm_R;
    pc2 -> norm_A = pc -> norm_A;
    pc2 -> norm_S = pc -> norm_S;
    pc2 -> tlh_R = pc -> tlh_R;
    pc2 -> tlh_A = pc -> tlh_A;
    pc2 -> tlh_S = pc -> tlh_S;
    pc2 -> trh_R = pc -> trh_R;
    pc2 -> trh_A = pc -> trh_A;
    pc2 -> trh_S = pc -> trh_S;
    pc2 -> brh_R = pc -> brh_R;
    pc2 -> brh_A = pc -> brh_A;
    pc2 -> brh_S = pc -> brh_S;

```

```

    return pc2;
}

```

```

void Utility::GE_RAS_CenterNormal2Points(GE_PCMRA_HEADER_OBJ *pc_loc,
    GE_PCMRA_HEADER_OBJ *pc_phase, int *xx1, int *yy1, int *xx2, int *yy2)
{

```

```

    float c1, c2, x1, x2, y1, y2;
    float pi = 3.141592654;
    float fov = pc_phase -> dfov;
    float fov_half = fov / 2.0;
    float sita = acos (pc_phase -> norm_R);

```

```

    Utility_Math *u = new Utility_Math();

```

```

    float dx, dy;

```

```

    if(sita <= pi/2.0)
    {
        dx = fov_half * cos (pi/2.0 - sita);
        dy = fov_half * sin (pi/2.0 - sita);
    }
    else if(sita > pi/2.0)
    {
        dx = fov_half * cos (sita - pi/2.0);
        dy = fov_half * sin (sita - pi/2.0);
    }

```

```

    printf("Utility:: fov = %f    sita = %f (%f)    dx dy %f %f\n", fov, sita,

```

```

pc_phase -> norm_R, dx, dy);

if(sita <= pi/2.0)
{
    x2 = pc_phase -> ctr_R - dx;
    x1 = pc_phase -> ctr_R + dx;
}
else if(sita > pi/2.0)
{
    pc_phase -> ctr_R - dx;
    pc_phase -> ctr_R + dx;
}

if(x1 > pc_loc -> tlh_R) x1 = pc_loc -> tlh_R;
else if(x1 < pc_loc -> trh_R) x1 = pc_loc -> trh_R;

if(x2 > pc_loc -> tlh_R) x2 = pc_loc -> tlh_R;
else if(x2 < pc_loc -> trh_R) x2 = pc_loc -> trh_R;

if(pc_loc->loc_ras == 'A' || pc_loc->loc_ras == 'P')
{
    y1 = pc_phase -> ctr_S - dy;
    y2 = pc_phase -> ctr_S + dy;

    if(y1 > pc_loc -> tlh_S) y1 = pc_loc -> tlh_S;
    else if(y1 < pc_loc -> brh_S) y1 = pc_loc -> brh_S;

    if(y2 > pc_loc -> tlh_S) y2 = pc_loc -> tlh_S;
    else if(y2 < pc_loc -> brh_S) y2 = pc_loc -> brh_S;

    printf(" RAS_LOC: A    %f %f\n", pc_loc -> tlh_S, pc_loc -> brh_S);

    u -> lineParaFromTwoPoints(pc_loc -> tlh_S, 0, pc_loc -> brh_S,
        (float)(pc_loc -> img_height), &c1, &c2);
    *yy1 = (int)(c1 * y1 + c2);
    *yy2 = (int)(c1 * y2 + c2);
}
else if(pc_loc->loc_ras == 'S' || pc_loc->loc_ras == 'I')
{
    y1 = pc_phase -> ctr_A - dy;
    y2 = pc_phase -> ctr_A + dy;

    if(y1 > pc_loc -> tlh_A) y1 = pc_loc -> tlh_A;
    else if(y1 < pc_loc -> brh_A) y1 = pc_loc -> brh_A;

    if(y2 > pc_loc -> tlh_A) y2 = pc_loc -> tlh_A;
    else if(y2 < pc_loc -> brh_A) y2 = pc_loc -> brh_A;

    printf(" RAS_LOC: S    %f %f\n", pc_loc -> tlh_A, pc_loc -> brh_A);

    u -> lineParaFromTwoPoints(pc_loc -> tlh_A, 0, pc_loc -> brh_A,
        (float)(pc_loc -> img_height), &c1, &c2);
    *yy1 = (int)(c1 * y1 + c2);
    *yy2 = (int)(c1 * y2 + c2);
}

printf(" RRR:    %f %f\n", pc_loc -> tlh_R, pc_loc -> trh_R);
u -> lineParaFromTwoPoints(pc_loc -> tlh_R, 0, pc_loc -> trh_R,
    (float)(pc_loc -> img_width), &c1, &c2);
*xx1 = (int)(c1 * x1 + c2);
*xx2 = (int)(c1 * x2 + c2);

printf(" x1 y1: %f %f  x2 y2:  %f %f \n", x1, y1, x2, y2);
printf(" xx1 yy1: %d %d  xx2, yy2: %d %d \n", *xx1, *yy1, *xx2, *yy2);

delete u;

```

```

//
alpha1 = get_angle(xc, point[i1].x, point[i1].y);
alpha2 = get_angle(xc, point[i2].x, point[i2].y);

//
// Find the point im2 (i.e., i1 or i2) so that
// the given angle "sita" is between the angle of the line
// connecting the "point im" and the center AND the the angle of
// the line connecting the point im2 and the center
//
if(sita >= alpha_min)
{
    if(sita <= alpha1)
        im2 = i1;
    else if(sita <= alpha2)
        im2 = i2;
    else if(alpha1 >= alpha2)
        im2 = i2;
    else
        im2 = i1;
}
else
{
    if(sita >= alpha1)
        im2 = i1;
    else if(sita >= alpha2)
        im2 = i2;
    else if(alpha1 >= alpha2)
        im2 = i1;
    else
        im2 = i2;
}

//
// Find the joint point of the two lines :
// (1) the line that passes the center and has angle "sita"
// (2) the line connecting the "point im" and the "point im2"
//
float x1 = point[im].x;
float y1 = point[im].y;
float x2 = point[im2].x;
float y2 = point[im2].y;

float a1, b1, a2, b2;

Utility_Math *u = new Utility_Math();

int flag1 = u -> lineParaFromTwoPoints(x1, y1, x2, y2, &a1, &b1);
int flag2 = u -> lineParaFromTwoPoints(xc, yc, sita, &a2, &b2);

if(flag1 == 0)
{
    if(flag2 == 0)
    {
        *x = x1;
        *y = y1;
    }
    else
    {
        *x = a1;
        *y = a2 * a1 + b2;
    }
}
else if(flag2 == 0)
{
    *x = a2;

```



```

        *y = a1 * a2 + b1;
    }
    else
    {
        if(fabsf(a1-a2) < 1.e-10)
        {
            *x = x1;
            *y = y1;
        }
        else
        {
            *x = - (b2-b1)/(a2-a1);
            *y = a1 * (*x) + b1;
        }
    }

    delete u;
}

```

```

void Utility::get_point(int num, Point *point, float *x, float *y)
{
    float sumx = 0;
    float sumy = 0;

    for(int i=0; i<num; i++)
    {
        sumx += point[i].x;
        sumy += point[i].y;
    }
    *x = sumx / (float)num;
    *y = sumy / (float)num;
}

```

```

FlowPara *Utility::get_flow(int w, int h, short **img, float pixel_area,
    unsigned char **mask, unsigned char **back)
{
    int    k = 0;
    float  vfr = 0;
    float  mv = 0;

    float  min_I = 1.0e20;
    float  max_I = -1.0e20;
    float  val;
    float  mean = 0;

    //printf("get_flow:: %d %d  %f\n", w, h, pixel_area);

    if(back != NULL)
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
    if(back[i][j] == 1)
    {
        val = float(img[i][j]);
        mean = mean + val;
        ++k;
    }

    if(k != 0) mean /= float(k);

    k = 0;
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
    if(mask[i][j] == 1)
    {
        val = float(img[i][j]) - mean;
        //printf(" Velocity [%d %d] = %f\n", j, i, val);
    }
}

```

```

    mv = mv + val/10.0;
    if(val > max_I) max_I = val;
    if(val < min_I) min_I = val;
    ++k;
}

float psv=0, bsv=0;

if(fabsf(min_I) > fabsf(max_I)) {psv = min_I / 10.0; bsv = max_I/10.0;}
else {psv = max_I / 10.0; bsv = min_I/10.0;}

vfr = (vfr) * 60.0;
if(k != 0) mv = (mv) / (float)k;
float area = pixel_area * k;

FlowPara *flow = new FlowPara;

/*
int maxmeide = int(area * 60) + 1;
for(int imeide=0; imeide<maxmeide; imeide++)
    if(area) printf(" imeide=%d area=%f\n", imeide, area);
*/

flow -> vfr = vfr;
flow -> psv = psv;
flow -> bsv = bsv;
flow -> mv = mv;
flow -> area = area;

return flow;
}

```

User: meide
Host: phoenix
Class: phoenix
Job: Utility_Vision.C

```
#include <stdio.h>
#include <malloc.h>
extern int Global_Error
```

65

```
unsigned char **alloc_img (int xsize,int ysize)
```

```
{ int i ;
  unsigned char **img, *img1 ;

  if(!(img = (unsigned char **)malloc( (long)ysize * sizeof(img1) ))) {
    fprintf(stderr,"Sorry, Computer getting stingy on memory (img) y =%d \n",
            ysize) ;
    return(NULL) ;
  }
  if(!(img1 = (unsigned char *)malloc( (long) ysize * xsize ))) {
    fprintf (stderr, "Sorry, Computer stingy on memory (img1) (%d,%d)\n",
            xsize, ysize) ;
    free(img) ; return(NULL) ;
  }
  for (i=0; i< ysize; i++) img[i] = &(img1[i * xsize]) ;
  return(img) ;
}
```

```
void free_img (unsigned char **img )
```

```
{
  free (*img) ;
  free (img) ;
}
```

```
float **alloc_fimg (int xsize,int ysize)
```

```
{ int i ;
  float **img ;

  if(!(img = (float **)malloc( (long)ysize * sizeof(float *))) {
    fprintf(stderr,"Sorry, Computer getting stingy on memory (fimg) y =%d \n",
            ysize) ;
    return(NULL) ;
  }
  if(!(*img = (float *)malloc( (long) ysize * xsize * sizeof(float)))) {
    fprintf (stderr, "Sorry, Computer stingy on memory (img1) (%d,%d)\n",
            xsize, ysize) ;
    free(img) ; return(NULL) ;
  }
  for (i=0; i< ysize; i++) img[i] = *img + (i * xsize) ;
  return(img) ;
}
```

```
void free_fimg (float **img )
```

```
{
  free (*img) ;
  free (img) ;
}
```

```
unsigned long **alloc_ulimg (int xsize,int ysize)
```

```
{ int i ;
  unsigned long **img ;

  if(!(img = (unsigned long **)malloc( (long)ysize * sizeof(unsigned long *))) {
    fprintf(stderr,"Sorry, Computer getting stingy on memory (fimg) y =%d \n",
            ysize) ;
    return(NULL) ;
  }
  if(!(*img = (unsigned long *)malloc( (long) ysize * xsize * sizeof(unsigned long))))
    fprintf (stderr, "Sorry, Computer stingy on memory (img1) (%d,%d)\n",
            xsize, ysize) ;
}
```

```

    free(img) ; return(NULL) ;
}
for (i=0; i< ysize; i++) img[i] = *img + (i * xsize) ;
return(img) ;
}

void free_ulimg (unsigned long **img )
{
    free (*img) ;
    free (img) ;
}

short **alloc_shimg (int xsize,int ysize)
{
    int i ;
    short **img ;

    if(!(img = (short **)malloc( (long)ysize * sizeof(short *)))) {
        fprintf(stderr,"Sorry, Computer getting stingy on memory (fimg) y=%d \n",
            ysize) ;
        return(NULL) ;
    }
    if(!(*img = (short *)malloc( (long) ysize * xsize * sizeof(short)))) {
        fprintf (stderr, "Sorry, Computer stingy on memory (img1) (%d,%d)\n",
            xsize, ysize) ;
        free(img) ; return(NULL) ;
    }
    for (i=0; i< ysize; i++) img[i] = *img + (i * xsize) ;
    return(img) ;
}

void free_shimg (short **img )
{
    free (*img) ;
    free (img) ;
}

```

```

/*
**      1st Version :   Darren Thompson
**      Reasearch Assistant, Biomedical Visualization Laboratory
**      Neurosurgery (M/C 799)
**      556N NPI
**      912 South Wood Street, Room 556
**      Chicago, IL USA   60612-7249
**      Phone: (312)996-9225
**      darren@spaldeholtz.bvl.uic.edu
*/

/*
**      2nd Version:      Meide Zhao,  Ph.D.
**      Director,  R&D of CANVAS Group
**      Dept. of Neurosugery
**      University of Illinois at Chicago
**      mzhao@uic.edu
*/

#include "GE.h"
#include "ImgAlloc.h"

short **Get_Img_Body(CANVAS_OBJ *canvas, int, GE_PCMRA_HEADER_OBJ *pc);
void Get_Img_Body2(CANVAS_OBJ *canvas, int, GE_PCMRA_HEADER_OBJ *pc,
    short **img);
void read_main(char *fname, CANVAS_OBJ *canvas, GE_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc);
void Get_Histogram_Info(CANVAS_OBJ *canvas, GE_HISTO_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc);
void Get_CT_Header(CANVAS_OBJ *canvas, GE_CT_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc);
void Get_MR_Header(CANVAS_OBJ *canvas, GE_MR_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc);

short **read_GE_CT_MRI(char *fname, GE_PCMRA_HEADER_OBJ *pc)
{
    CANVAS_OBJ *canvas = new CANVAS_OBJ;
    GE_HEADER_OBJ *ge = new GE_HEADER_OBJ;
    GE_HISTO_HEADER_OBJ *ge_histo = new GE_HISTO_HEADER_OBJ;
    GE_CT_HEADER_OBJ *ge_ct = new GE_CT_HEADER_OBJ;
    GE_MR_HEADER_OBJ *ge_mr = new GE_MR_HEADER_OBJ;
    short **img;

    read_main(fname, canvas, ge, pc);

    Get_Histogram_Info(canvas, ge_histo, pc);
    canvas -> filePosition = ge->img_p_image;

    if (ge->img_l_image == MR_HEADER_SIZE)
        Get_MR_Header(canvas, ge_mr, pc);
    else if (ge->img_l_image == CT_HEADER_SIZE)
        Get_CT_Header(canvas, ge_ct, pc);

    img = Get_Img_Body(canvas, ge->img_hdr_length, pc);

    fclose(canvas -> fp);

    delete canvas;
    delete ge;
    delete ge_histo;
    delete ge_ct;
    delete ge_mr;

    return(img);
}

/* reader_GE_MRI_CT */

```

```

void read_GE(char *fname, GE_PCMRA_HEADER_OBJ *pc, short img)
{
    CANVAS_OBJ *canvas = new CANVAS_OBJ;
    GE_HEADER_OBJ *ge = new GE_HEADER_OBJ;
    GE_HISTO_HEADER_OBJ *ge_histo = new GE_HISTO_HEADER_OBJ;
    GE_CT_HEADER_OBJ *ge_ct = new GE_CT_HEADER_OBJ;
    GE_MR_HEADER_OBJ *ge_mr = new GE_MR_HEADER_OBJ;

    read_main(fname, canvas, ge, pc);

    Get_Histogram_Info(canvas, ge_histo, pc);
    canvas -> filePosition = ge->img_p_image;

    if (ge->img_l_image == MR_HEADER_SIZE)
        Get_MR_Header(canvas, ge_mr, pc);
    else if (ge->img_l_image == CT_HEADER_SIZE)
        Get_CT_Header(canvas, ge_ct, pc);

    Get_Img_Body2(canvas, ge->img_hdr_length, pc, img);

    fclose(canvas -> fp);

    delete canvas;
    delete ge;
    delete ge_histo;
    delete ge_ct;
    delete ge_mr;
}

/* reader_GE_MRI_CT */

void read_main(char *fname, CANVAS_OBJ *canvas, GE_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc)
{
    short i=1;
    char ImgId[80];

    canvas -> filePosition = 0;
    canvas -> numberOfBytesRead = 0;

    canvas -> fp = fopen(fname, "r");

    if(canvas -> fp == NULL)
    {
        printf("\n Can't open file : %s\n\n", fname);
        exit(0);
    }

    fread(&ge->img_magic, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);

    fread(&ge->img_hdr_length, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);

    fread(&ge->img_width, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);

    fread(&ge->img_height, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);

    fread(&ge->img_depth, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);

    fread(&ge->img_compress, sizeof(int), 1, canvas -> fp);
    canvas -> numberOfBytesRead += sizeof(int);
}

```

```

fread(&ge->img_dwing, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_dlevel, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_bgshade, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_ovrflow, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_undflow, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_top_offset, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_bot_offset, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_version, 2, 1, canvas -> fp);
canvas -> numberOfBytesRead += 2;

fread(&ge->img_p_id, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_id, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_unpack, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_unpack, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_compress, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_compress, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_histo, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_histo, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_text, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_text, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_graphics, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_graphics, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_dbHdr, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

```



```

fread(&ge->img_l_dbH, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_levelOffset, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_user, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_user, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_suite, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_suite, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_exam, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_exam, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_series, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_series, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_p_image, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fread(&ge->img_l_image, sizeof(int), 1, canvas -> fp);
canvas -> numberOfBytesRead += sizeof(int);

fgetpos(canvas -> fp, &canvas -> filePosition);

canvas -> filePosition = ge->img_p_histo;
fsetpos(canvas -> fp, &canvas -> filePosition);

pc->img_width = ge->img_width;
pc->img_height = ge->img_height;
}

void Get_Histogram_Info(CANVAS_OBJ *canvas, GE_HISTO_HEADER_OBJ *ge,
GE_PCMRA_HEADER_OBJ *pc)
{
    short i=0;

    fread(&ge->hs_version, 4, 1, canvas -> fp);

    fread(&ge->hs_sd, 4, 1, canvas -> fp);

    fread(&ge->hs_mean, 2, 1, canvas -> fp);

    fread(&ge->hs_min, 2, 1, canvas -> fp);

    fread(&ge->hs_max, 2, 1, canvas -> fp);

    fread(&ge->hs_first, 2, 1, canvas -> fp);

    fread(&ge->hs_region, 2, 1, canvas -> fp);

```

```

fread(&ge->hs_length, 2, 1, canvas -> fp);
for (i=0; i<ge->hs_length; i++)
{
    fread((ge->hs_bins+i), 2, 1, canvas -> fp);
}

pc->hs_min = ge->hs_min;
pc->hs_max = ge->hs_max;

return;

}/* Get_Histogram_Info */

void Get_CT_Header(CANVAS_OBJ *canvas, GE_CT_HEADER_OBJ *ge,
GE_PCMRA_HEADER_OBJ *pc)
{
    fpos_t headerOffset = canvas -> filePosition;

    canvas -> filePosition += 12;
    fseekpos(canvas -> fp, &canvas -> filePosition);

    fread(&ge->im_no, 2, 1, canvas -> fp);

    canvas -> filePosition += 14;
    fseekpos(canvas -> fp, &canvas -> filePosition);

    fread(&ge->slthick, 4, 1, canvas -> fp);

    fread(&ge->imatrix_X, 2, 1, canvas -> fp);

    fread(&ge->imatrix_Y, 2, 1, canvas -> fp);

    fread(&ge->dfov, 4, 1, canvas -> fp);

    fread(&ge->dfov_rect, 4, 1, canvas -> fp);

    fread(&ge->dim_X, 4, 1, canvas -> fp);

    fread(&ge->dim_Y, 4, 1, canvas -> fp);

    fread(&ge->pixsize_X, 4, 1, canvas -> fp);

    fread(&ge->pixsize_Y, 4, 1, canvas -> fp);

    canvas -> filePosition = headerOffset + 114;
    fseekpos(canvas -> fp, &canvas -> filePosition);

    fread(&ge->planes, 2, 1, canvas -> fp);

    canvas -> filePosition = headerOffset + 116;
    fseekpos(canvas -> fp, &canvas -> filePosition);

    fread(&ge->scanspacing, 4, 1, canvas -> fp);

    fgetpos(canvas -> fp, &canvas -> filePosition);

    pc->slthick = ge->slthick;
    pc->imatrix_X = ge->imatrix_X;
    pc->imatrix_Y = ge->imatrix_Y;

    pc->dim_X = ge->dim_X;
    pc->dim_Y = ge->dim_Y;

```

```

pc->dfov_rect = ge->dfov_rect;

pc->dfov = ge->dfov;
pc->pixsize_X = ge->pixsize_X;
pc->pixsize_Y = ge->pixsize_Y;
pc->scanspacing = ge->scanspacing;

return;

}/* Get_CT_Header */

short **Get_Img_Body(CANVAS_OBJ *canvas, int img_hdr_length, GE_PCMRA_HEADER_OBJ *pc)
{
    fpos_t tmpPosition;
    short **img;
    short *p;

    tmpPosition = img_hdr_length;
    fseekpos(canvas -> fp, &tmpPosition);

    img = NULL;
    img = (short **)alloc_shimg(pc->img_width, (pc->img_height + 1));
    p = *img;

    while (!feof(canvas -> fp))
    {
        fread(p, 2, 1, canvas -> fp);
        ++p;
    }
    return(img);
}

void Get_Img_Body2(CANVAS_OBJ *canvas, int img_hdr_length,
    GE_PCMRA_HEADER_OBJ *pc, short **img)
{
    fpos_t tmpPosition;
    short *p;

    tmpPosition = img_hdr_length;
    fseekpos(canvas -> fp, &tmpPosition);

    p = *img;

    while (!feof(canvas -> fp))
    {
        fread(p, 2, 1, canvas -> fp);
        ++p;
    }
    for (int i=0; i< pc->img_height; i++)
        img[i] = *img + (i * pc->img_width) ;
}

void Get_MR_Header(CANVAS_OBJ *canvas, GE_MR_HEADER_OBJ *ge,
    GE_PCMRA_HEADER_OBJ *pc)
{
    fpos_t headerOffset = canvas -> filePosition;

    canvas -> filePosition += 12;
    fseekpos(canvas -> fp, &canvas -> filePosition);

    fread(&ge->im_no, 2, 1, canvas -> fp);

    canvas -> filePosition += 14;
    fseekpos(canvas -> fp, &canvas -> filePosition);
}

```

```

fread(&ge->slthick, 1, canvas -> fp);
fread(&ge->imatrix_X, 2, 1, canvas -> fp);
fread(&ge->imatrix_Y, 2, 1, canvas -> fp);
fread(&ge->dfov, 4, 1, canvas -> fp);
fread(&ge->dfov_rect, 4, 1, canvas -> fp);
fread(&ge->dim_X, 4, 1, canvas -> fp);
fread(&ge->dim_Y, 4, 1, canvas -> fp);
fread(&ge->pixsize_X, 4, 1, canvas -> fp);
fread(&ge->pixsize_Y, 4, 1, canvas -> fp);

canvas -> filePosition = headerOffset+114;
fsetpos(canvas -> fp, &canvas -> filePosition);

fread(&ge->planes, 2, 1, canvas -> fp);

canvas -> filePosition = headerOffset+116;
fsetpos(canvas -> fp, &canvas -> filePosition);

fread(&ge->scanspacing, 4, 1, canvas -> fp);

/* Meide's Stuff Begin */
fread(&ge->img_compress, 2, 1, canvas -> fp);
fread(&ge->img_scouttype, 2, 1, canvas -> fp);
fread(&ge->loc_ras, 1, 1, canvas -> fp);
fread(&ge->tmp, 1, 1, canvas -> fp);
fread(&ge->loc, 4, 1, canvas -> fp);
fread(&ge->ctr_R, 4, 1, canvas -> fp);
fread(&ge->ctr_A, 4, 1, canvas -> fp);
fread(&ge->ctr_S, 4, 1, canvas -> fp);

fread(&ge->norm_R, 4, 1, canvas -> fp);
fread(&ge->norm_A, 4, 1, canvas -> fp);
fread(&ge->norm_S, 4, 1, canvas -> fp);

fread(&ge->tlh_R, 4, 1, canvas -> fp);
fread(&ge->tlh_A, 4, 1, canvas -> fp);
fread(&ge->tlh_S, 4, 1, canvas -> fp);

fread(&ge->trh_R, 4, 1, canvas -> fp);
fread(&ge->trh_A, 4, 1, canvas -> fp);
fread(&ge->trh_S, 4, 1, canvas -> fp);

```

```

fread(&ge->brh_R, 4, 1, canvas -> fp);
fread(&ge->brh_A, 4, 1, canvas -> fp);
fread(&ge->brh_S, 4, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 194;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->tr, 4, 1, canvas -> fp);
fread(&ge->ti, 4, 1, canvas -> fp);
fread(&ge->te, 4, 1, canvas -> fp);
fread(&ge->te2, 4, 1, canvas -> fp);

fread(&ge->num_echo, 2, 1, canvas -> fp);
fread(&ge->echo_num, 2, 1, canvas -> fp);

fread(&ge->table_delta, 4, 1, canvas -> fp);
fread(&ge->num_excitations, 4, 1, canvas -> fp);
fread(&ge->continuous_slice_flag, 2, 1, canvas -> fp);
fread(&ge->heart_rate, 2, 1, canvas -> fp);
fread(&ge->delay_time, 4, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 242;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->cardiac_rep_time, 4, 1, canvas -> fp);
fread(&ge->num_img_per_cardiac_cycle, 2, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 254;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->pc_flow_axis, 2, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 574;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->pc_flow_axis, 2, 1, canvas -> fp);
fread(&ge->pc_venc, 2, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 584;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->img_type, 2, 1, canvas -> fp);

canvas -> filePosition = headerOffset + 646;
fsetpos(canvas -> fp, &canvas -> filePosition);
fread(&ge->cardiac_phase_num, 2, 1, canvas -> fp);

```

```

canvas -> filePosition = headerOffset + 728;
fsetpos(canvas -> fp, &canvas -> filePosition);

fread(&ge->scan_acquisition_no, 2, 1, canvas -> fp);

fread(&ge->mag_wighting_flag, 2, 1, canvas -> fp);

fread(&ge->venc_weighted_scale, 4, 1, canvas -> fp);


canvas -> filePosition = headerOffset + 738;
fsetpos(canvas -> fp, &canvas -> filePosition);

fread(&ge->num_of_phases, 4, 1, canvas -> fp);


pc->slthick = ge->slthick;
pc->imatrix_X = ge->imatrix_X;
pc->imatrix_Y = ge->imatrix_Y;


pc->dim_X = ge->dim_X;
pc->dim_Y = ge->dim_Y;
pc->dfov_rect = ge->dfov_rect;


pc->dfov = ge->dfov;
pc->pixsize_X = ge->pixsize_X;
pc->pixsize_Y = ge->pixsize_Y;
pc->scanspacing = ge->scanspacing;


pc->tr = ge->tr;
pc->te = ge->te;


pc->num_excitations = ge->num_excitations;
pc->heart_rate = ge->heart_rate;
pc->delay_time = ge->delay_time;


pc->num_img_per_cardiac_cycle = ge->num_img_per_cardiac_cycle;
pc->flip_angle = ge->flip_angle;
pc->pc_flow_axis = ge->pc_flow_axis;
pc->pc_venc = ge->pc_venc;
pc->cardiac_phase_num = ge->cardiac_phase_num;
pc->num_of_phases = ge->num_of_phases;


pc->mag_weighting_flag = ge->mag_wighting_flag & 2;
pc->venc_weighted_scale = ge->venc_weighted_scale;


pc->loc_ras = ge->loc_ras;


pc->ctr_R = ge->ctr_R;
pc->ctr_A = ge->ctr_A;
pc->ctr_S = ge->ctr_S;


printf(" RAS %f %f %f \n", pc->ctr_R, pc->ctr_A, pc->ctr_S);


pc->norm_R = ge->norm_R;
pc->norm_A = ge->norm_A;
pc->norm_S = ge->norm_S;


pc->tlh_R = ge->tlh_R;
pc->tlh_A = ge->tlh_A;
pc->tlh_S = ge->tlh_S;


pc->trh_R = ge->trh_R;
pc->trh_A = ge->trh_A;
pc->trh_S = ge->trh_S;


pc->brh_R = ge->brh_R;

```

```
pc->brh_A = ge->brh_A;  
pc->brh_S = ge->brh_S
```

76

```
/* Meide's Stuff End */
```

```
return;
```

```
}/* Get_MR_Header */
```

```
#include "ImgBase.h"
#include "ImgAlloc.h"
```

77

```
#include <math.h>
#include <stdio.h>
#include <unistd.h>
```

```
ImgBase::ImgBase()
{
    imgdata = NULL;
}
```

```
ImgBase::ImgBase(int w, int h, short **img)
{
    width = w;
    height = h;
    imgdata = img;
}
```

```
ImgBase::~ImgBase()
{
    if(imgdata != NULL) { free_shimg(imgdata); imgdata = NULL; }
}
```

```
void ImgBase::set_imgdata(short **img)
{
    if (imgdata != NULL)
    { free_shimg(imgdata); imgdata = NULL; }
    imgdata = img;
}
```

```
short **ImgBase::get_imgdata()
{
    return imgdata;
}
```



```

#include "ImgGE.h"
#include "ImgAlloc.h"
#include <math.h>
#include <Vk/VkComponent.h>
#include <stdio.h>
#include "GE.h"
#include <unistd.h>
#include "Utility.h"
#include "Utility_Vision.h"

```

```

ImgGE::ImgGE(char *fname) : ImgBase()
{
    pc = new GE_PCMRA_HEADER_OBJ;
    imgdata = read_GE_CT_MRI(fname, pc);
    width = pc->img_width;
    height = pc->img_height;
    //printf("ImgGE::  %f %f %f\n", pc->slthick, pc->pixsize_X, pc->pixsize_Y);
    //printf("ImgGE::  %f %d %d\n", pc->dfov, pc->imatrix_X, pc->imatrix_Y);
}

```

```

ImgGE::ImgGE() : ImgBase()
{
    pc = NULL;
}

```

```

ImgGE::~~ImgGE()
{
    if(pc != NULL) delete pc;
    if(imgdata != NULL) { free_shimg(imgdata); imgdata = NULL; }
}

```

```

void ImgGE::set(char *fname)
{
    if(pc != NULL)
    {
        delete pc;
        pc = NULL;
    }
    if(imgdata != NULL)
    {
        free_shimg(imgdata);
        imgdata = NULL;
    }
    pc = new GE_PCMRA_HEADER_OBJ;
    imgdata = read_GE_CT_MRI(fname, pc);
    width = pc->img_width;
    height = pc->img_height;
}

```

```

void ImgGE::inverseImg()
{
    for(int y=0; y<height; y++)
        for(int x=0; x<width; x++)
            imgdata[y][x] = -imgdata[y][x];
}

```

```

ImgGE *ImgGE::copy()
{
    ImgGE *ie = new ImgGE();
    int w = this -> get_width();
    int h = this -> get_height();
    ie -> set_width(w);
    ie -> set_height(h);

    Utility_Vision *uv = new Utility_Vision();
    Utility *u = new Utility();
}

```

```

    ie -> set_imgdata( u -> copy_img(w, h, this->get_imgdata()) );

    ie -> set_header( u -> copy_pc(this->get_header()) );

    delete uv;
    delete u;
    return ie;
}

void ImgGE::set(ImgGE *ie)
{
    if(pc != NULL)
    {
        delete pc;
        pc = NULL;
    }

    if(imgdata != NULL)
    {
        free_shimg(imgdata);
        imgdata = NULL;
    }

    set_header(ie -> get_header());

    set_width(ie -> get_width());
    set_height(ie -> get_height());

    set_imgdata(ie -> get_imgdata());
}

GE_PCMRA_HEADER_OBJ *ImgGE::get_header()
{
    return pc;
}

void ImgGE::set(float zoom, short **img, int x, int y, int w, int h)
{
    int x1 = int( (float)x/zoom );
    int y1 = int( (float)y/zoom );
    int w1 = int( (float)w/zoom );
    int h1 = int( (float)h/zoom );

    printf(" ImgGE::get_ROI  %d %d  %d %d\n", x1, y1, w1, h1);

    width = w1;
    height = h1;
    set_imgdata(get_ROI(x1, y1, w1, h1, img));
}

void ImgGE::set(int x, int y, int w, int h, short **img)
{
    width = w;
    height = h;
    set_imgdata(get_ROI(x, y, w, h, img));
}

void ImgGE::set(int x, int y, int w, int h, short **img, unsigned char **mask, float ra)
{
    width = w;
    height = h;
    set_imgdata(get_ROI(x, y, w, h, img, mask, ratio));
}

```

```

void ImgGE::set(unsigned char **mask, float in_ratio)
{
    int w1 = width;
    int h1 = height;

    float val, mm;
    int i, j;

    float min_I = 1.0e30;
    float max_I = -1.0e30;

    for(i=0; i<h1; i++)
    for(j=0; j<w1; j++)
        if(mask[i][j] == 1)
        {
            val = float(imgdata[i][j]);
            if(val < min_I) min_I = val;
            if(val > max_I) max_I = val;
        }

    min_I = fabsf(min_I);
    max_I = fabsf(max_I);

    float ratio = in_ratio/100.0;

    printf("\n*****\n\n %f %f    Ratio:  %f\n",
        min_I, max_I, ratio);

    float mm1;
    if(min_I > max_I) mm1 = min_I;
    else mm1 = max_I;

    if(mm1 > 600.0) mm1 = 600.0;
    mm = mm1 * ratio;

    /*
    if(min_I > max_I) mm = min_I * ratio;
    else mm = max_I * ratio;
    */

    for(i=0; i<h1; i++)
    for(j=0; j<w1; j++)
        if(mask[i][j] == 0 )
        {
            val = fabsf(imgdata[i][j]);
            if(val >= mm) imgdata[i][j] = 0;
        }
        else if(mask[i][j] == 1)
        {
            val = fabsf(imgdata[i][j]);
            if(val > 600.0 && imgdata[i][j] > 0 ) imgdata[i][j] = 0;
            if(val > 600.0 && imgdata[i][j] < 0 ) imgdata[i][j] = -600;
        }
    }

short **ImgGE::get_ROI(int x1, int y1, int w1, int h1, short **img)
{
    short **img2 = (short **)alloc_shimg(w1,h1);
    for(int i=0; i<h1; i++)
    for(int j=0; j<w1; j++)
        img2[i][j] = img[y1+i][x1+j];
    return img2;
}

```

```

short **ImgGE::get_ROI(int x1, int y1, int w1, int h1,
    short **img, unsigned char **mask, float in_ratio)
{
    short **img2 = (short **)alloc_shimg(w1,h1);
    float val, mm;
    int i, j;

    float min_I = 1.0e30;
    float max_I = -1.0e30;

    for(i=0; i<h1; i++)
    for(j=0; j<w1; j++)
        if(mask[i][j] == 1)
        {
            img2[i][j] = img[y1+i][x1+j];
            val = float(img2[i][j]);
            if(val < min_I) min_I = val;
            if(val > max_I) max_I = val;
        }

    min_I = fabsf(min_I);
    max_I = fabsf(max_I);

    float ratio = in_ratio/100.0;

    printf(" Ratio:  %f\n", ratio);

    float mm1;
    if(min_I > max_I) mm1 = min_I;
    else mm1 = max_I;

    if(mm1 > 600.0) mm1 = 600.0;
    mm = mm1 * ratio;
    /*
    if(min_I > max_I) mm = min_I * ratio;
    else mm = max_I * ratio;
    */
    for(i=0; i<h1; i++)
    for(j=0; j<w1; j++)
        if(mask[i][j] == 0 )
        {
            val = fabsf(img[y1+i][x1+j]);
            if(val >= mm) img2[i][j] = 0;
            else img2[i][j] = img[y1+i][x1+j];
        }
        else if(mask[i][j] == 1)
        {
            val = fabsf(img[y1+i][x1+j]);
            if(val > 600.0 && img[y1+i][x1+j] > 0 ) img2[i][j] = 0;
            if(val > 600.0 && img[y1+i][x1+j] < 0 ) img2[i][j] = -600;
        }
    return img2;
}

unsigned char **ImgGE::thresh(int x, int y, int w, int h, float low)
{
    unsigned char **img = (unsigned char **)alloc_img(w,h);
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
        if(float(imgdata[y+i][x+j]) >= low) img[i][j] = 1;
        else img[i][j] = 0;
    return img;
}

```

//---- Start editable code block: headers and declarations

#include <stdio.h>

#include "ImgGE.h"

#include "BbUI.h"

#include "Utility.h"

#include "BbDisplay.h"

#include "BbFlow.h"

#include "Flow.h"

#include <Xm/ScrolledW.h>

#include "BbHistogram.h"

#include "BbRHistogram.h"

#include "BbLWaveform.h"

#include "BbRWaveform.h"

#include "HistoTwoLinesDrawingArea.h"

#include "LineDrawingArea.h"

#include "BbLROI.h"

#include "BbRROI.h"

#include "BbFormat.h"

#include "ROI.h"

#include "Animate.h"

#include "Utility_Widget.h"

#include <sys/types.h>

#include <unistd.h>

#include "Utility_Vision.h"

#include "Utility_Math.h"

#include "Progress.h"

#include "ProgressMainWindow.h"

#include "Utility_3D.h"

#include "Win3DMainWindow.h"

#include "snake.h"

#include "GS_Points.h"

//---- End editable code block: headers and declarations

//---- ObjectManager Constructor

ObjectManager::ObjectManager()
{

//---- Start editable code block: ObjectManager constructor 2

//---- End editable code block: ObjectManager constructor 2

} // End Constructor

ObjectManager::~ObjectManager()

```

{
    //----- Start editable code block: ObjectManager destructor

    //----- End editable code block: ObjectManager destructor

}    // End Destructor

void ObjectManager::init()
{
    _img = NULL;
    _imgView = NULL;

    _img2 = NULL;
    _imgView2 = NULL;
    _imgViewLoc = NULL;

    _histoView = NULL;
    _histoView2 = NULL;

    _waveView = NULL;
    _waveView2 = NULL;

    _L3D = NULL;
    _R3D = NULL;
    _win3D = NULL;
    _root = NULL;

    _animate = NULL;
    _map = NULL;

    progress = NULL;
    _patients = NULL;

    _magImg = NULL;
    _phaImg = NULL;

    msgsLeft.img_space = IMAGE_2D;
    msgsLeft.img_pcmra_type = PCMRA_MAGNITUDE;

    msgsLeft.user = USER_NOVIES;

    msgsLeft.layout = LAYOUT_NORMAL;
    msgsLeft.loc_x1 = -1;

    msgsLeft.img_zoom = 1.0;
    msgsLeft.img_scale_type = SCALE_SPLINE;
    msgsLeft.img_zoom_select = ZOOM_BOTH;

    msgsLeft.img_visual_type = VISUAL_GRAY;

    msgsLeft.flow_select = FLOW_VFR;

    msgsLeft.roi_action = ROI_REDEFINE;
    msgsLeft.roi_type = ROI_RECTANGLE;

    msgsLeft.roi_mode = ROI_LEFT;

    msgsLeft.histo_status = HISTOGRAM_COARSE;

    msgsLeft.show_status = TRUE;
    msgsLeft.img_space = IMAGE_2D;

    msgsLeft.postThresh = 0.0;

```

```

msgsLeft.negThresh = 0.0;
msgsLeft.magThresh = 0;

msgsRight.show_status = TRUE;
msgsRight.img_space = IMAGE_2D;

msgsRight.flow_select = FLOW_PSV;
msgsRight.flow_method = FLOW_MANUAL;

msgsRight.velocity_select = VELOCITY_ASIS;
msgsRight.velocity_ratio = 50.0;

msgsRight.img_select = RIGHT_IMG_REF;
msgsRight.img_pcmra_type = PCMRA_MAGNITUDE;

msgsRight.ratio3D = 0;
msgsRight.camera = CAMERA_ORTHO;
msgsRight.Fixed3D = 0;

msgsRight.img_number_prev = -1;

msgsRight.img_zoom = 1.0;
msgsRight.img_scale_type = SCALE_SPLINE;

msgsRight.img_visual_type = VISUAL_GRAY;

msgsRight.roi_action = ROI_REDEFINE;
msgsRight.roi_type = ROI_RECTANGLE;

msgsRight.histo_status = HISTOGRAM_COARSE;

msgsRight.roi_changed = 1;
msgsRight.roi_mask = NULL;
msgsRight.roi_flow = NULL;
msgsRight.roi_back = NULL;
msgsRight.roi_points = NULL;

msgsRight.show_detail = FALSE;
msgsRight.flow_noiseLevel = 2;

msgsRight.num_cardiacs = 3;

msgsRight.publish = PUBLISH_NONE;
msgsRight.flow3DDir = 1;
msgsRight.HR = 70.0;

msgsRight.animate_mode = ANIMATE_2D;

sprintf(msgsRight.pubDir, "/usr/people/canvas/active_patients");
}

//---- End of generated code

//---- Start editable code block: End of generated code

//
// Given:
//      new img_number
//      new PCMRA image type
//
// Update:
//      _img _imgView _histoview
//      _img2 _imgView2 _histoview2 _flow
//

```

```

void ObjectManager::update_Limg(int img_number)
{
    printf("\n\nupdate_Limg\n");
    set_Llowhigh();
    update_Limg(img_number);
    printf("\n\nupdate_Rimg\n");
    update_Rimg(img_number);
}

void ObjectManager::get_general()
{
    _img = get_ImgGE(msgsLoaded.img_start, msgsLeft.img_type, PCMRA_PHASE, _img);
    Utility *u = new Utility();
    _GE_header = u -> copy_pc(_img -> get_header());
    delete u;
    if(_GE_header -> heart_rate > 0)
    {
        msgsRight.HR = _GE_header -> heart_rate;
    }
    delete _img;
    _img = NULL;
}

void ObjectManager::update_Limg(int img_number)
{
    msgsLeft.img_number = img_number;

    Utility_Widget *u = new Utility_Widget();

    u->set_textfield(((BbDisplay *) (_LDisp))->textfieldDisplayImgNumber, img_number);
    u->set_label(((BbUI *) (_bb))->labelImgNumber, img_number);

    _img = get_ImgGE(img_number, msgsLeft.img_type, msgsLeft.img_pcmra_type, _img);

    printf("\n\n update_LimgView\n");
    update_LimgView();

    delete u;
}

void ObjectManager::update_LimgView()
{
    if(msgsLeft.img_space == IMAGE_2D)
    {
        hideL3D();
        showL2D();
        update_LimgView2D();
    }
    else if(msgsLeft.img_space == IMAGE_3D)
    {
        hideL2D();
        _img2 = get_ImgGE2(msgsRight.img_number, (ImgGE *) _img2);
        update_LimgView3D();
    }
}

//
// Used for changing img_number and/or changing zoom factor
//
void ObjectManager::update_LimgView2D()
{
    int w1, h1, w2, h2;
    Boolean p, c;
}

```



```

if(_imgView != NULL)
{
    p = get_LscaleSize(_imgView -> _zoom, &w2, &h2);
    c = get_LscaleSize(msgsLeft.img_zoom, &w1, &h1);
}

if(p) printf(" Current _imgView TRUE\n");
if(c) printf(" Future _imgView TRUE\n");

if(_imgView != NULL && ( (!c && !p) || (p && c && w1 <= w2 && h1 <= h2) ) )
{
    //
    // The imgsize is under control
    //
    _imgView -> set(_img->get_width(), _img->get_height(), _img->get_imgdata(),
        msgsLeft.img_visual_type, msgsLeft.img_scale_type,
        msgsLeft.img_zoom, msgsLeft.img_winCenter, msgsLeft.img_winWidth, msgsRight.flc);

    int xc = 316;
    int yc = 346;
    if(msgsLeft.layout == LAYOUT_NORMAL)
    {
        ((DrawingArea *)_imgView) -> set-Origin(xc - int(float(w1)/2.0), yc - int(float(h1)/2.0));
        printf(" Left Origin:: %d %d\n", xc - int(float(w1)/2.0), yc - int(float(h1)/2.0));
        _imgView -> display();
        if(_imgViewLoc != NULL)
        {
            _imgViewLoc -> hide();
        }
    }
    else if(msgsLeft.layout == LAYOUT_COMBO)
    {
        ((DrawingArea *)_imgView) -> set-Origin(xc - int(float(w1)/2.0), yc - 256 - 20);
        //printf(" Left Origin:: %d %d\n", xc - int(float(w1)/2.0), yc - int(float(h1)/2.0));
        _imgView -> display();

        if(_imgViewLoc != NULL)
        {
            _imgViewLoc -> show();
            ((DrawingArea *)_imgViewLoc) -> display(xc-128, yc);
            _imgViewLoc -> update(msgsRight.lowGrayRef, msgsRight.highGrayRef);

            if(msgsLeft.loc_x1 > 0)
            {
                Utility_Widget *uw = new Utility_Widget();
                Widget wid = _imgViewLoc -> baseWidget();
                GC gc = uw -> get_xorGC(wid);
                XDrawLine(XtDisplay(wid), XtWindow(wid), gc,
                    msgsLeft.loc_x1, msgsLeft.loc_y1,
                    msgsLeft.loc_x2, msgsLeft.loc_y2);
                XtReleaseGC(wid, gc);
                delete uw;
            }
        }
    }
}
else
{
    if(_imgView != NULL)
        delete _imgView;

    new_LimgView();
}

if(_imgView -> _zoomImg != NULL) update_Lhisto();

```

```

}

void ObjectManager::update_LimgView(float center, float width)
{
    if(msgsLeft.img_space == IMAGE_2D)
    {
        update_Llowhigh();
        _imgView -> update(center, width);
        if(_histoView != NULL)
        {
            ((HistoTwoLinesDrawingArea *)_histoView) -> update_lowhigh(center, width);
            ((HistoTwoLinesDrawingArea *)_histoView) -> change();
        }
    }
}

Boolean ObjectManager::get_LscaleSize(float zoom, int *w, int *h)
{
    int w1 = _img->get_width();
    int h1 = _img->get_height();

    Utility_Math *u = new Utility_Math();

    *w = u->int_t(w1 * zoom);
    *h = u->int_t(h1 * zoom);

    delete u;

    if(*w > LEFT_MAX_WIDTH || *h > LEFT_MAX_HEIGHT) return TRUE;
    else return FALSE;
}

void ObjectManager::new_LimgView()
{
    int w = _img->get_width();
    int h = _img->get_height();

    int w2;
    int h2;

    int xc = 316;
    int yc = 346;
    int x0 = 60;
    int y0 = 90;

    if(get_LscaleSize(msgsLeft.img_zoom, &w2, &h2))
    {
        _imgView = new ROI MedDrawingArea("GE", _bb->baseWidget(), 1);
        _imgView -> setObj(this);
        _imgView -> set(w, h, _img->get_imgdata(), msgsLeft.img_visual_type, msgsLeft.in
msgsLeft.img_zoom, msgsLeft.img_winCenter, msgsLeft.img_winWidth, msgsRight.flow
        _imgView -> show();
        ((DrawingArea *)_imgView) -> display(x0, y0, LEFT_MAX_WIDTH, LEFT_MAX_HEIGHT);
    }
    else
    {
        _imgView = new ROI MedDrawingArea("GE", _bb->baseWidget(), 0);
        _imgView -> setObj(this);
        _imgView -> set(w, h, _img->get_imgdata(), msgsLeft.img_visual_type, msgsLeft.img
msgsLeft.img_zoom, msgsLeft.img_winCenter, msgsLeft.img_winWidth, msgsRight.flow
        _imgView -> show();
        ((DrawingArea *)_imgView) -> display(xc-w2/2, yc-h2/2);
    }
}

```

```

_imgView -> _roi_type = msgsLeft.roi_type;
_imgView -> _roi_act = msgsLeft.roi_action;
}

```

```

void ObjectManager::update_Lhisto()
{

```

```

    int w = _imgView->get_width();
    int h = _imgView->get_height();
    short **img = _imgView->_zoomImg;

```

```

    int dw = 400;
    int dh = 80;

```

```

    if(_histoView == NULL)
    {

```

```

        printf("  New HistoDrawingArea  %d %d\n", w, h);
        _histoView = new HistoTwoLinesDrawingArea(dw, dh,
            "Lhisto", _LHist -> baseWidget() );
        _histoView -> set(_LHist->_labelLHistoMin,
            _LHist->_labelLHistoMax,
            _LHist->_labelLHistoLow,
            _LHist->_labelLHistoHigh);
        ((HistoTwoLinesDrawingArea *)_histoView) -> set(this, MY_LEFT);
        ((HistoTwoLinesDrawingArea *)_histoView) -> set(w, h, img, dw);
        ((HistoTwoLinesDrawingArea *)_histoView) -> newTwoLines(msgsLeft.img_winCenter,
            msgsLeft.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView) -> display(0, 35);
        ((HistoTwoLinesDrawingArea *)_histoView) -> show();
    }

```

```

    else
    {

```

```

        ((HistoTwoLinesDrawingArea *)_histoView) -> set(w, h, img, dw);
        ((HistoTwoLinesDrawingArea *)_histoView) -> newTwoLines(msgsLeft.img_winCenter,
            msgsLeft.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView) -> display();
        ((HistoTwoLinesDrawingArea *)_histoView) -> _twolines -> draw();
    }
}

```

```

void ObjectManager::update_Lhisto2()
{

```

```

    int w = _imgView->get_width();
    int h = _imgView->get_height();
    short **img = _imgView->_zoomImg;

```

```

    int dw = 400;
    int dh = 80;

```

```

    if(_histoView == NULL)
    {

```

```

        printf("  New HistoDrawingArea  %d %d\n", w, h);
        _histoView = new HistoTwoLinesDrawingArea(dw, dh,
            "Lhisto", _LHist -> baseWidget() );
        _histoView -> set(_LHist->_labelLHistoMin,
            _LHist->_labelLHistoMax,
            _LHist->_labelLHistoLow,
            _LHist->_labelLHistoHigh);
        ((HistoTwoLinesDrawingArea *)_histoView) -> set(this, MY_LEFT);
        ((HistoTwoLinesDrawingArea *)_histoView) -> set(w, h, img, dw, NULL,
            msgsLeft.img_winCenter-200.0, msgsLeft.img_winWidth+200.0 );
        ((HistoTwoLinesDrawingArea *)_histoView) -> newTwoLines(msgsLeft.img_winCenter,
            msgsLeft.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView) -> display(0, 35);
        ((HistoTwoLinesDrawingArea *)_histoView) -> show();
    }
}

```

```

}
else
{
    ((HistoTwoLinesDrawingArea *)_histoView) -> set(w, h, img, dw, NULL,
        msgsLeft.img_winCenter-200.0, msgsLeft.img_winWidth+200.0 );
    ((HistoTwoLinesDrawingArea *)_histoView) -> newTwoLines(msgsLeft.img_winCenter,
        msgsLeft.img_winWidth);
    ((HistoTwoLinesDrawingArea *)_histoView) -> display();
    ((HistoTwoLinesDrawingArea *)_histoView) -> _twolines -> draw();
}
}

void ObjectManager::update_Rimg(int img_number)
{
    msgsRight.img_number = img_number;

    _img2 = get_ImgGE2(img_number, (ImgGE *)_img2);

    set_Rlowhigh();

    update_RimgView();

    //if(msgsRight.img_type == IMAGE_PCMRA && msgsRight.img_select == RIGHT_IMG_ROI)
    //{
    //    ((BbLROI *)_LROI) -> initROI();
    //}
}

void ObjectManager::update_RimgView()
{
    int x1, y1, w1, h1;

    if(msgsRight.img_space == IMAGE_2D)
    {
        hide3D();
        show2D();
        if(_imgView2 != NULL &&
            msgsRight.img_select == RIGHT_IMG_ROI &&
            msgsRight.img_type == IMAGE_PCMRA &&
            msgsRight.img_pcmra_type == PCMRA_VELOCITY &&
            msgsRight.velocity_select == VELOCITY_FLOWMASKED)
        {
            update_Rimg2D();

            x1 = int( float(msgsLeft.roi_x)/_imgView->_zoom );
            y1 = int( float(msgsLeft.roi_y)/_imgView->_zoom );
            w1 = int( float(msgsLeft.roi_w)/_imgView->_zoom );
            h1 = int( float(msgsLeft.roi_h)/_imgView->_zoom );

            printf(" velocity_ratio %f\n", msgsRight.velocity_ratio);

            if(msgsRight.roi_mask != NULL)
                _img2 -> set(x1, y1, w1, h1, _img -> get_imgdata(),
                    msgsRight.roi_mask, msgsRight.velocity_ratio);
            else
                _img2 -> set(x1, y1, w1, h1, _img -> get_imgdata());
        }
        update_RimgView2D();
    }
    else if(msgsRight.img_space == IMAGE_3D)
    {
        hide2D();

        if(_imgView2 != NULL &&
            msgsRight.img_select == RIGHT_IMG_ROI &&

```

```

msgsRight.img_type == IMAGE_PCMRA &&
msgsRight.img_pcm_type == PCMRA_VELOCITY &&
msgsRight.velocity_select == VELOCITY_FLOWMASKED)
{
    update_Rimg2D();

    x1 = int( float(msgsLeft.roi_x)/_imgView->_zoom );
    y1 = int( float(msgsLeft.roi_y)/_imgView->_zoom );
    w1 = int( float(msgsLeft.roi_w)/_imgView->_zoom );
    h1 = int( float(msgsLeft.roi_h)/_imgView->_zoom );

    if(msgsRight.roi_mask != NULL)
        _img2 -> set(x1, y1, w1, h1, _img -> get_imgdata(),
            msgsRight.roi_mask, msgsRight.velocity_ratio);
    else
        _img2 -> set(x1, y1, w1, h1, _img -> get_imgdata());
}

update_RimgView3D();
}

if(_imgView2 -> _ROI != NULL)
{
    if(msgsRight.flow_method == FLOW_AUTOSNAKE)
    {
        GS_Points *p0 = new GS_Points();

        int i, step;

        Points *p = &(_imgView2 -> _ROI -> _points_in_border);
        if(p -> _numPoints < 100) step = 1;
        else if(p -> _numPoints < 200) step = 2;
        else if(p -> _numPoints < 300) step = 3;
        else if(p -> _numPoints < 400) step = 4;
        else step = 5;

        for(i=0; i< p -> _numPoints; i += step)
            p0 -> add( p -> _points[i].x, p -> _points[i].y );

        int row = _imgView2->get_height();
        int col = _imgView2->get_width();
        float **fimg = _imgView2->getFloatImg();

        printf("\n Snake Initial Points ==> %d \n", p0 -> _numPoints);

        GS_Points *p1 = snake(row, col, fimg, p0);

        p -> clear();
        for(i=0; i< p1 -> _numPoints; i++)
            p -> add(p1 -> _points[i].x, p1 -> _points[i].y );
        printf("\n Snake Points generated ==> %d \n", p1 -> _numPoints);
    }

    _imgView2 -> AcceptROI();
    msgsRight.roi_flow = _imgView2 -> _ROI -> copyArea();
    update_flow();
}

}

void ObjectManager::update_Rimg2D()
{
    _imgView2 -> setData(_img2->get_width(), _img2->get_height(), _img2->get_imgdata(),
        msgsRight.img_visual_type, msgsRight.img_scale_type,

```

```
msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.  
91
```

```
update_mask();
```

```
}
```

```
void ObjectManager::update_mask()
```

```
{
```

```
if(msgsRight.img_type == IMAGE_PCMRA &&  
    msgsRight.img_select == RIGHT_IMG_ROI &&  
    msgsRight.img_pcmra_type == PCMRA_VELOCITY)
```

```
{
```

```
if(_imgView2 -> _ROI != NULL && _imgView2 -> _ROI -> _areaOrg != NULL)
```

```
{
```

```
int w1 = _img2 -> get_width();  
int h1 = _img2 -> get_height();  
short **img1 = _img2 -> get_imgdata();  
GE_PCMRA_HEADER_OBJ *pc = _img->get_header();
```

```
if(msgsRight.roi_mask != NULL)  
    free_img(msgsRight.roi_mask);
```

```
msgsRight.roi_mask = alloc_img(w1, h1);
```

```
int i, j, i1, j1;  
float zoom = msgsRight.img_zoom;
```

```
int i0 = int(zoom / 2.0);
```

```
printf(" update_mask %d %d %f \n", w1, h1, zoom);
```

```
if(msgsRight.flow_method == FLOW_MANUAL)
```

```
{
```

```
for(i=0; i<h1; i++)
```

```
for(j=0; j<w1; j++)
```

```
{
```

```
    msgsRight.roi_mask[i][j] = _imgView2 -> _ROI -> _areaOrg[i][j];
```

```
}
```

```
}
```

```
else if(msgsRight.flow_method == FLOW_SEMIAUTO)
```

```
{
```

```
get_minmaxFlow(1);
```

```
printf(" update_mask %f %f\n", _imgView2 -> _minFlow, _imgView2 -> _maxF
```

```
for(i=0; i<h1; i++)
```

```
for(j=0; j<w1; j++)
```

```
{
```

```
    if(_imgView2 -> _ROI -> _areaOrg[i][j] == 1)
```

```
{
```

```
        if(img1[i][j] >= _imgView2 -> _minFlow &&
```

```
            img1[i][j] <= _imgView2 -> _maxFlow)
```

```
            msgsRight.roi_mask[i][j] = 1;
```

```
        else msgsRight.roi_mask[i][j] = 0;
```

```
}
```

```
    else msgsRight.roi_mask[i][j] = 0;
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
void ObjectManager::update_Rimg2D(ImgGE *ie)
```

```
{
```

```
_imgView2 -> setData(ie->get_width(), ie->get_height(), ie->get_imgdata(),
```

```

msgsRight.img_visual_type, msgsRight.img_scale_t,
msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.

```

```

update_mask(ie);

```

```

}

```

```

void ObjectManager::update_mask(ImgGE *ie)

```

```

{

```

```

    if(msgsRight.img_type == IMAGE_PCMRA &&
        msgsRight.img_select == RIGHT_IMG_ROI &&
        msgsRight.img_pcmra_type == PCMRA_VELOCITY)
    {

```

```

        if(_imgView2 -> _ROI != NULL && _imgView2 -> _ROI -> _areaOrg != NULL)
        {

```

```

            int w1 = ie -> get_width();
            int h1 = ie -> get_height();
            short **img1 = ie -> get_imgdata();
            GE_PCMRA_HEADER_OBJ *pc = ie->get_header();

```

```

            if(msgsRight.roi_mask != NULL)
                free_img(msgsRight.roi_mask);

```

```

            msgsRight.roi_mask = alloc_img(w1, h1);

```

```

            int i, j, i1, j1;
            float zoom = msgsRight.img_zoom;

```

```

            int i0 = int(zoom / 2.0);

```

```

            printf(" update_mask %d %d %f \n", w1, h1, zoom);

```

```

            if(msgsRight.flow_method == FLOW_MANUAL)
            {

```

```

                for(i=0; i<h1; i++)
                for(j=0; j<w1; j++)
                {
                    msgsRight.roi_mask[i][j] = _imgView2 -> _ROI -> _areaOrg[i][j];
                }
            }

```

```

            else if(msgsRight.flow_method == FLOW_SEMIAUTO)
            {

```

```

                get_minmaxFlow(1);

```

```

                printf(" update_mask %f %f\n", _imgView2 -> _minFlow, _imgView2 -> _maxF

```

```

                for(i=0; i<h1; i++)
                for(j=0; j<w1; j++)
                {

```

```

                    if(_imgView2 -> _ROI -> _areaOrg[i][j] == 1)
                    {
                        if(img1[i][j] >= _imgView2 -> _minFlow &&
                            img1[i][j] <= _imgView2 -> _maxFlow)
                            msgsRight.roi_mask[i][j] = 1;
                        else msgsRight.roi_mask[i][j] = 0;
                    }
                    else msgsRight.roi_mask[i][j] = 0;
                }
            }

```

```

        }

```

```

    }

```

```

}

```

```

void ObjectManager::update_RimgView2D()

```

```

{

```

```

    int w1, h1, w2, h2;
    Boolean p, c;

```

```

imgGE -> set(_imgView -> _zoom, imgGE -> get_imgdata(),
    msgslLeft.roi_x, msgslLeft.roi_y, msgslLeft.roi_w, msgslLeft.roi_h );
//printf(" Flow::\n");
pc = imgGE -> get_header();
img1 = u -> ToVelocity(pc, w1, h1, _img2->get_imgdata(),
    imgGE->get_imgdata(), msgslLeft.posThresh,
    msgslLeft.negThresh, msgslLeft.magThresh);
//printf(" Flow:: 1\n");
delete imgGE;
}
else if(msgsRight.img_pcmra_type == PCMRA_PHASE)
{
    char fname[300];
    int tmp = msgsLoaded.img_start2 + (msgsRight.img_number - msgsLoaded.in
    sprintf(fname, "%s/E%dS%dI%d.MR", msgsLoaded.img_dir, msgsLoaded.img_ex
        msgsLoaded.img_series, tmp);

    ImgGE *imgGE = new ImgGE(fname);
    printf(" Flow::PCMRA_PHASE fname=%s\n", fname);

    printf(" Flow::PCMRA_PHASE %d %d\n", imgGE->get_width(), imgGE->get_hei

    imgGE -> set(_imgView -> _zoom, imgGE -> get_imgdata(),
        msgslLeft.roi_x, msgslLeft.roi_y, msgslLeft.roi_w, msgslLeft.roi_h );

    //printf(" Flow::\n");
    pc = _img2->get_header();
    img1 = u -> ToVelocity(pc, w1, h1, imgGE -> get_imgdata(),
        _img2->get_imgdata(), msgslLeft.posThresh,
        msgslLeft.negThresh, msgslLeft.magThresh);

    //printf(" Flow:: 1\n");
    delete imgGE;
}

//float tmpX = pc->dim_X / float(pc->img_width);
//float tmpY = pc->dim_Y / float(pc->img_height);
//printf(" tmpX tmpY %f %f %f %f %f %f\n", tmpX, tmpY, pc->dim_X, pc
// pc->img_width, pc->img_height);
float pixel_area = pc->pixsize_x * pc->pixsize_y; // 10000;
//pixel_area = pixel_area * tmpX * tmpY;

int w2;
int h2;

w2 = int((float)_imgView2->get_width()/_imgView2->_zoom);
h2 = int((float)_imgView2->get_height()/_imgView2->_zoom);

unsigned char **area_img = alloc_img(w2, h2);
unsigned char **back_img = NULL;

int i, j, i1, j1;

int i0 = int(_imgView2->_zoom / 2.0);

if(msgsRight.roi_back == NULL)
    back_img = NULL;
else
{
    back_img = alloc_img(w2, h2);
    for(i=0; i<h2; i++)
        for(j=0; j<w2; j++)
        {
            i1 = int( (float)i * _imgView2->_zoom ) + i0;

```



```

        j1 = int( (float)j * _imgView2->_zoom ) + i0;
        back_img[i1][j] = msgsRight.roi_back[i1][j1];
    }
}

if(msgsRight.flow_method == FLOW_MANUAL || msgsRight.flow_method == FLOW_F
{
    for(i=0; i<h2; i++)
        for(j=0; j<w2; j++)
        {
            i1 = int( (float)i * _imgView2->_zoom ) + i0;
            j1 = int( (float)j * _imgView2->_zoom ) + i0;
            area_img[i][j] = area_flow[i1][j1];
        }
}
else if(msgsRight.flow_method == FLOW_SEMIAUTO)
{
    if(msgsRight.img_space == IMAGE_2D)
    {
        get_minmaxFlow();
    }
    else if(msgsRight.img_space == IMAGE_3D)
        get_minmaxFlow(1);

    for(i=0; i<h2; i++)
        for(j=0; j<w2; j++)
        {
            i1 = int( (float)i * _imgView2->_zoom ) + i0;
            j1 = int( (float)j * _imgView2->_zoom ) + i0;

            if(area_flow[i1][j1] == 1)
            {
                if(_imgView2->_zoomImg[i1][j1] >= _imgView2->_minFlow &&
                   _imgView2->_zoomImg[i1][j1] <= _imgView2->_maxFlow &&
                   img1[i][j] >= _imgView2->_minFlow && img1[i][j] <= _imgView2
                   area_img[i][j] = 1;
                else area_img[i][j] = 0;
            }
            else area_img[i][j] = 0;
        }
    }
}

//flow = flow -> get_flow(w1, h1, img1, psv, area, area_img, back_img);
//area_img(area_img);

int im = msgsRight.img_number - msgsLoaded.img_start;
int vessel = _vessel;

/* printf("\n ROI:: %f : %d %d %d %d\n", _imgView->_zoom,
        msgsLeft.roi_x, msgsLeft.roi_y, msgsLeft.roi_w, msgsLeft.roi_h);*/

printf("\nFLOW :: Vessel: %d Phase: %d --> %f %f %f %f %f\n",
        vessel, im, flow->vfr, flow->mv, flow->psv, flow->bsv, flow->area);

_flow[vessel].vesselFlows[im].vfr = flow->vfr;
_flow[vessel].vesselFlows[im].psv = flow->psv;
_flow[vessel].vesselFlows[im].bsv = flow->bsv;
_flow[vessel].vesselFlows[im].mv = flow->mv;
_flow[vessel].vesselFlows[im].area = flow->area;
//_flow[vessel].numPoints = im + 1;

uw -> set_textfield(((BbFlow *)_RFlow) -> _textfieldVFR, um->int_t(flow->v
uw -> set_textfield(((BbFlow *)_RFlow) -> _textfieldPSV, um->int_t(flow->p
uw -> set_textfield(((BbFlow *)_RFlow) -> _textfieldBSV, um->int_t(flow->k

```

```

uw -> set_testfield((BbFlow *)_RFlow) -> _testfieldMV, um->int_t(flow->mv
uw -> set_testfield((BbFlow *)_RFlow) -> _testfieldArea, float(flow->area

```

```

delete flow;

```

```

}

```

```

}
delete u;
delete um;
delete uw;
delete uv;
}

```

```

void ObjectManager::get_minmaxFlow(int flag)

```

```

{
float minI, maxI;
_imgView2 -> get_mmFlow(msgsRight.roi_flow, &minI, &maxI);

if(msgsRight.flowDir < 0)
{
maxI = minI * float(msgsRight.flow_noiseLevel)/100.0;
_RFlow -> set_noiseLevel(msgsRight.flow_noiseLevel, maxI);
}
else if(msgsRight.flowDir > 0)
{
minI = maxI * float(msgsRight.flow_noiseLevel)/100.0;
_RFlow -> set_noiseLevel(msgsRight.flow_noiseLevel, minI);
}

if(flag == 0)
_imgView2 -> semiFlow(minI, maxI, msgsRight.roi_flow);
else
_imgView2 -> semiFlow2(minI, maxI, msgsRight.roi_flow);
}

```

```

void ObjectManager::saveFlow()

```

```

{
FILE *fp = fopen("flow.dat", "w");
fprintf(fp, "Patient\n");
fprintf(fp, "Anatomy\n");
//fprintf(fp, "%d\n", heart_rate);
fprintf(fp, "78\n");

int vessel = _vessel;

for(int i=0; i<msgsRight.num_imgs; i++)
fprintf(fp, "%d %f %f %f %f\n", i, _flow[vessel].vesselFlows[i].vfr, _flow[vessel]
_flow[vessel].vesselFlows[i].mv, _flow[vessel].vesselFlows[i].area);

fclose(fp);
}

```

```

void ObjectManager::localizer()

```

```

{
GE_PCMRA_HEADER_OBJ *pc_phase = _img -> get_header();
GE_PCMRA_HEADER_OBJ *pc_loc = _img2 -> get_header();

Utility *u = new Utility();
Utility_Math *um = new Utility_Math();
Utility_Widget *uw = new Utility_Widget();

int x1, y1, x2, y2;

u -> GE_RAS_CenterNormal2Points(pc_loc, pc_phase,

```

```
float zoom = msgsRight.img_zoom;
x1 = um -> int_t(zoom * float(x1));
y1 = um -> int_t(zoom * float(y1));
x2 = um -> int_t(zoom * float(x2));
y2 = um -> int_t(zoom * float(y2));
```

```
Widget wid = _imgView2 -> baseWidget();
GC gc = uw -> get_xorGC(wid);
```

```
XDrawLine(XtDisplay(wid), XtWindow(wid), gc,
          x1, y1, x2, y2);
```

```
msgsLeft.loc_x1 = x1;
msgsLeft.loc_y1 = y1;
msgsLeft.loc_x2 = x2;
msgsLeft.loc_y2 = y2;
```

```
XtReleaseGC(wid, gc);
delete u;
delete um;
delete uw;
```

```
}
```

```
void ObjectManager::update_Llowhigh()
```

```
{
    if(msgsLeft.img_type == IMAGE_PCMRA)
    {
        if(msgsLeft.img_pcmra_type == PCMRA_MAGNITUDE)
        {
            msgsLeft.lowMag = msgsLeft.img_winCenter;
            msgsLeft.highMag = msgsLeft.img_winWidth;
            printf(" update_Llowhigh:: Mag %f %f\n", msgsLeft.lowMag, msgsLeft.highMag);
        }
        else if(msgsLeft.img_pcmra_type == PCMRA_PHASE)
        {
            msgsLeft.lowPha = msgsLeft.img_winCenter;
            msgsLeft.highPha = msgsLeft.img_winWidth;
        }
        else if(msgsLeft.img_pcmra_type == PCMRA_VELOCITY)
        {
            msgsLeft.low = msgsLeft.img_winCenter;
            msgsLeft.high = msgsLeft.img_winWidth;
            printf(" update_Llowhigh:: Vel %f %f\n", msgsLeft.low, msgsLeft.high);
        }
    }
    else
    {
        msgsLeft.low = msgsLeft.img_winCenter;
        msgsLeft.high = msgsLeft.img_winWidth;
    }
}
```

```
void ObjectManager::set_Llowhigh()
```

```
{
    if(msgsLeft.img_type == IMAGE_PCMRA)
    {
        if(msgsLeft.img_pcmra_type == PCMRA_MAGNITUDE)
        {
            msgsLeft.img_winCenter = msgsLeft.lowMag;
            msgsLeft.img_winWidth = msgsLeft.highMag;
            printf(" set_Llowhigh:: Mag %f %f\n", msgsLeft.lowMag, msgsLeft.highMag);
        }
        else if(msgsLeft.img_pcmra_type == PCMRA_PHASE)
```

```

{
    msgsLeft.img_winCenter = msgsLeft.lowPha;
    msgsLeft.img_winWidth = msgsLeft.highPha;
}
else if(msgsLeft.img_pcmra_type == PCMRA_VELOCITY)
{
    msgsLeft.img_winCenter = msgsLeft.low;
    msgsLeft.img_winWidth = msgsLeft.high;
    printf(" set_Llowhigh:: Vel %f %f\n", msgsLeft.low, msgsLeft.high);
}
}
else
{
    msgsLeft.img_winCenter = msgsLeft.low;
    msgsLeft.img_winWidth = msgsLeft.high;
}
}

void ObjectManager::update_Rlowhigh()
{
    if(msgsRight.img_type == IMAGE_PCMRA)
    {
        if(msgsRight.img_pcmra_type == PCMRA_MAGNITUDE)
        {
            if(msgsRight.img_visual_type == VISUAL_GRAY)
            {
                switch(msgsRight.img_select)
                {
                    case RIGHT_IMG_WHOLE:
                        msgsRight.lowMagGrayWhole = msgsRight.img_winCenter;
                        msgsRight.highMagGrayWhole = msgsRight.img_winWidth;
                        printf(" \n\nupdate_Rlowhigh:: Mag %f %f\n", msgsRight.lowMagGrayWhole,
                            break;
                    case RIGHT_IMG_ROI:
                        msgsRight.lowMagGrayROI = msgsRight.img_winCenter;
                        msgsRight.highMagGrayROI = msgsRight.img_winWidth;
                        break;
                    case RIGHT_IMG_REF:
                        msgsRight.lowMagGrayRef = msgsRight.img_winCenter;
                        msgsRight.highMagGrayRef = msgsRight.img_winWidth;
                        break;
                    case RIGHT_IMG_OTHER:
                        msgsRight.lowMagGrayOther = msgsRight.img_winCenter;
                        msgsRight.highMagGrayOther = msgsRight.img_winWidth;
                        break;
                    default:
                        msgsRight.low = msgsRight.img_winCenter;
                        msgsRight.high = msgsRight.img_winWidth;
                        break;
                }
            }
        }
        else if(msgsRight.img_visual_type == VISUAL_COLOR)
        {
            switch(msgsRight.img_select)
            {
                case RIGHT_IMG_WHOLE:
                    msgsRight.lowMagColorWhole = msgsRight.img_winCenter;
                    msgsRight.highMagColorWhole = msgsRight.img_winWidth;
                    break;
                case RIGHT_IMG_ROI:
                    msgsRight.lowMagColorROI = msgsRight.img_winCenter;
                    msgsRight.highMagColorROI = msgsRight.img_winWidth;
                    break;
                case RIGHT_IMG_REF:
                    msgsRight.lowMagColorRef = msgsRight.img_winCenter;
                    msgsRight.highMagColorRef = msgsRight.img_winWidth;

```

```

        break;
    case RIGHT_IMG_OTHER:
        msgsgRight.lowMagColorOther = msgsgRight.img_winCenter;
        msgsgRight.highMagColorOther = msgsgRight.img_winWidth;
        break;
    default:
        msgsgRight.low = msgsgRight.img_winCenter;
        msgsgRight.high = msgsgRight.img_winWidth;
        break;
    }
}
}
else if(msgsgRight.img_pcmra_type == PCMRA_PHASE)
{
    if(msgsgRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsgRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsgRight.lowPhaGrayWhole = msgsgRight.img_winCenter;
                msgsgRight.highPhaGrayWhole = msgsgRight.img_winWidth;
                break;
            case RIGHT_IMG_ROI:
                msgsgRight.lowPhaGrayROI = msgsgRight.img_winCenter;
                msgsgRight.highPhaGrayROI = msgsgRight.img_winWidth;
                break;
            case RIGHT_IMG_REF:
                msgsgRight.lowPhaGrayRef = msgsgRight.img_winCenter;
                msgsgRight.highPhaGrayRef = msgsgRight.img_winWidth;
                break;
            case RIGHT_IMG_OTHER:
                msgsgRight.lowPhaGrayOther = msgsgRight.img_winCenter;
                msgsgRight.highPhaGrayOther = msgsgRight.img_winWidth;
                break;
            default:
                msgsgRight.low = msgsgRight.img_winCenter;
                msgsgRight.high = msgsgRight.img_winWidth;
                break;
        }
    }
}
else if(msgsgRight.img_visual_type == VISUAL_COLOR)
{
    switch(msgsgRight.img_select)
    {
        case RIGHT_IMG_WHOLE:
            msgsgRight.lowPhaColorWhole = msgsgRight.img_winCenter;
            msgsgRight.highPhaColorWhole = msgsgRight.img_winWidth;
            break;
        case RIGHT_IMG_ROI:
            msgsgRight.lowPhaColorROI = msgsgRight.img_winCenter;
            msgsgRight.highPhaColorROI = msgsgRight.img_winWidth;
            break;
        case RIGHT_IMG_REF:
            msgsgRight.lowPhaColorRef = msgsgRight.img_winCenter;
            msgsgRight.highPhaColorRef = msgsgRight.img_winWidth;
            break;
        case RIGHT_IMG_OTHER:
            msgsgRight.lowPhaColorOther = msgsgRight.img_winCenter;
            msgsgRight.highPhaColorOther = msgsgRight.img_winWidth;
            break;
        default:
            msgsgRight.low = msgsgRight.img_winCenter;
            msgsgRight.high = msgsgRight.img_winWidth;
            break;
    }
}
}
}

```

```

}
else if(msgsRight.img_pcmra_type == PCMRA_VELOCITY)
{
    if(msgsRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsRight.lowGrayWhole = msgsRight.img_winCenter;
                msgsRight.highGrayWhole = msgsRight.img_winWidth;
                printf(" \n\nupdate_Rlowhigh:: Vel %f %f\n",msgsRight.lowGrayWhole, n
                break;
            case RIGHT_IMG_ROI:
                msgsRight.lowGrayROI = msgsRight.img_winCenter;
                msgsRight.highGrayROI = msgsRight.img_winWidth;
                break;
            case RIGHT_IMG_REF:
                msgsRight.lowGrayRef = msgsRight.img_winCenter;
                msgsRight.highGrayRef = msgsRight.img_winWidth;
                break;
            case RIGHT_IMG_OTHER:
                msgsRight.lowGrayOther = msgsRight.img_winCenter;
                msgsRight.highGrayOther = msgsRight.img_winWidth;
                break;
            default:
                msgsRight.low = msgsRight.img_winCenter;
                msgsRight.high = msgsRight.img_winWidth;
                break;
        }
    }
    else if(msgsRight.img_visual_type == VISUAL_COLOR)
    {
        switch(msgsRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsRight.lowColorWhole = msgsRight.img_winCenter;
                msgsRight.highColorWhole = msgsRight.img_winWidth;
                break;
            case RIGHT_IMG_ROI:
                msgsRight.lowColorROI = msgsRight.img_winCenter;
                msgsRight.highColorROI = msgsRight.img_winWidth;
                break;
            case RIGHT_IMG_REF:
                msgsRight.lowColorRef = msgsRight.img_winCenter;
                msgsRight.highColorRef = msgsRight.img_winWidth;
                break;
            case RIGHT_IMG_OTHER:
                msgsRight.lowColorOther = msgsRight.img_winCenter;
                msgsRight.highColorOther = msgsRight.img_winWidth;
                break;
            default:
                msgsRight.low = msgsRight.img_winCenter;
                msgsRight.high = msgsRight.img_winWidth;
                break;
        }
    }
}
}
else
{
    if(msgsRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsRight.lowGrayWhole = msgsRight.img_winCenter;

```

```

        msgsRight.highGrayWhole = msgsRight.img_winWidth;
        break;
    case RIGHT_IMG_ROI:
        msgsRight.lowGrayROI = msgsRight.img_winCenter;
        msgsRight.highGrayROI = msgsRight.img_winWidth;
        break;
    case RIGHT_IMG_REF:
        msgsRight.lowGrayRef = msgsRight.img_winCenter;
        msgsRight.highGrayRef = msgsRight.img_winWidth;
        break;
    case RIGHT_IMG_OTHER:
        msgsRight.lowGrayOther = msgsRight.img_winCenter;
        msgsRight.highGrayOther = msgsRight.img_winWidth;
        break;
    default:
        msgsRight.low = msgsRight.img_winCenter;
        msgsRight.high = msgsRight.img_winWidth;
        break;
}

```

```

}
else if(msgsRight.img_visual_type == VISUAL_COLOR)
{

```

```

    switch(msgsRight.img_select)
    {

```

```

        case RIGHT_IMG_WHOLE:
            msgsRight.lowColorWhole = msgsRight.img_winCenter;
            msgsRight.highColorWhole = msgsRight.img_winWidth;
            break;
        case RIGHT_IMG_ROI:
            msgsRight.lowColorROI = msgsRight.img_winCenter;
            msgsRight.highColorROI = msgsRight.img_winWidth;
            break;
        case RIGHT_IMG_REF:
            msgsRight.lowColorRef = msgsRight.img_winCenter;
            msgsRight.highColorRef = msgsRight.img_winWidth;
            break;
        case RIGHT_IMG_OTHER:
            msgsRight.lowColorOther = msgsRight.img_winCenter;
            msgsRight.highColorOther = msgsRight.img_winWidth;
            break;
        default:
            msgsRight.low = msgsRight.img_winCenter;
            msgsRight.high = msgsRight.img_winWidth;
            break;
    }
}

```

```

void ObjectManager::set_Rlowhigh()
{

```

```

    printf(" \n\n      set_Rlowhigh\n");
    if(msgsRight.img_type == IMAGE_PCMRA)
    {

```

```

        if(msgsRight.img_pcmra_type == PCMRA_MAGNITUDE)
        {

```

```

            if(msgsRight.img_visual_type == VISUAL_GRAY)
            {

```

```

                switch(msgsRight.img_select)
                {

```

```

                    case RIGHT_IMG_WHOLE:
                        msgsRight.img_winCenter = msgsRight.lowMagGrayWhole;
                        msgsRight.img_winWidth = msgsRight.highMagGrayWhole;
                        printf(" \n\n      set_Rlowhigh:: Mag %f %f\n", msgsRight.lowMagGrayWhole,

```

```

        break;
    case RIGHT_IMG_ROI:
        msgsRight.img_winCenter = msgsRight.lowMagGrayROI;
        msgsRight.img_winWidth = msgsRight.highMagGrayROI;
        break;
    case RIGHT_IMG_REF:
        msgsRight.img_winCenter = msgsRight.lowMagGrayRef;
        msgsRight.img_winWidth = msgsRight.highMagGrayRef;
        printf(" \n\n      set_Rlowhigh:: Mag REF\n");
        break;
    case RIGHT_IMG_OTHER:
        msgsRight.img_winCenter = msgsRight.lowMagGrayOther;
        msgsRight.img_winWidth = msgsRight.highMagGrayOther;
        break;
    default:
        msgsRight.img_winCenter = msgsRight.low;
        msgsRight.img_winWidth = msgsRight.high;
        break;
}
}
else if(msgsRight.img_visual_type == VISUAL_COLOR)
{
    switch(msgsRight.img_select)
    {
        case RIGHT_IMG_WHOLE:
            msgsRight.img_winCenter = msgsRight.lowMagColorWhole;
            msgsRight.img_winWidth = msgsRight.highMagColorWhole;
            break;
        case RIGHT_IMG_ROI:
            msgsRight.img_winCenter = msgsRight.lowMagColorROI;
            msgsRight.img_winWidth = msgsRight.highMagColorROI;
            break;
        case RIGHT_IMG_REF:
            msgsRight.img_winCenter = msgsRight.lowMagColorRef;
            msgsRight.img_winWidth = msgsRight.highMagColorRef;
            printf(" \n\n      set_Rlowhigh:: Mag REF\n");
            break;
        case RIGHT_IMG_OTHER:
            msgsRight.img_winCenter = msgsRight.lowMagColorOther;
            msgsRight.img_winWidth = msgsRight.highMagColorOther;
            break;
        default:
            msgsRight.img_winCenter = msgsRight.low;
            msgsRight.img_winWidth = msgsRight.high;
            break;
    }
}
}
else if(msgsRight.img_pcmra_type == PCMRA_PHASE)
{
    if(msgsRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsRight.img_winCenter = msgsRight.lowPhaGrayWhole;
                msgsRight.img_winWidth = msgsRight.highPhaGrayWhole;
                break;
            case RIGHT_IMG_ROI:
                msgsRight.img_winCenter = msgsRight.lowPhaGrayROI;
                msgsRight.img_winWidth = msgsRight.highPhaGrayROI;
                break;
            case RIGHT_IMG_REF:
                msgsRight.img_winCenter = msgsRight.lowPhaGrayRef;
                msgsRight.img_winWidth = msgsRight.highPhaGrayRef;
                break;
        }
    }
}

```



```

    case RIGHT_IMG_OTHER:
        msgsgRight.img_winCenter = msgsgRight.lowPhaGrayOther;
        msgsgRight.img_winWidth = msgsgRight.highPhaGrayOther;
        printf(" \n\n      set_Rlowhigh:: Pha REF\n");
        break;
    default:
        msgsgRight.img_winCenter = msgsgRight.low;
        msgsgRight.img_winWidth = msgsgRight.high;
        break;
}
}
else if(msgsgRight.img_visual_type == VISUAL_COLOR)
{
    switch(msgsgRight.img_select)
    {
        case RIGHT_IMG_WHOLE:
            msgsgRight.img_winCenter = msgsgRight.lowPhaColorWhole;
            msgsgRight.img_winWidth = msgsgRight.highPhaColorWhole;
            break;
        case RIGHT_IMG_ROI:
            msgsgRight.img_winCenter = msgsgRight.lowPhaColorROI;
            msgsgRight.img_winWidth = msgsgRight.highPhaColorROI;
            break;
        case RIGHT_IMG_REF:
            msgsgRight.img_winCenter = msgsgRight.lowPhaColorRef;
            msgsgRight.img_winWidth = msgsgRight.highPhaColorRef;
            printf(" \n\n      set_Rlowhigh:: Pha REF\n");
            break;
        case RIGHT_IMG_OTHER:
            msgsgRight.img_winCenter = msgsgRight.lowPhaColorOther;
            msgsgRight.img_winWidth = msgsgRight.highPhaColorOther;
            break;
        default:
            msgsgRight.img_winCenter = msgsgRight.low;
            msgsgRight.img_winWidth = msgsgRight.high;
            break;
    }
}
}
else if(msgsgRight.img_pcmra_type == PCMRA_VELOCITY)
{
    if(msgsgRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsgRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsgRight.img_winCenter = msgsgRight.lowGrayWhole;
                msgsgRight.img_winWidth = msgsgRight.highGrayWhole;
                printf(" \n\n      set_Rlowhigh:: Vel %f %f\n",msgsgRight.lowGrayWhol
                break;
            case RIGHT_IMG_ROI:
                msgsgRight.img_winCenter = msgsgRight.lowGrayROI;
                msgsgRight.img_winWidth = msgsgRight.highGrayROI;
                break;
            case RIGHT_IMG_REF:
                msgsgRight.img_winCenter = msgsgRight.lowGrayRef;
                msgsgRight.img_winWidth = msgsgRight.highGrayRef;
                printf(" \n\n      set_Rlowhigh:: Vel REF\n");
                break;
            case RIGHT_IMG_OTHER:
                msgsgRight.img_winCenter = msgsgRight.lowGrayOther;
                msgsgRight.img_winWidth = msgsgRight.highGrayOther;
                break;
            default:
                msgsgRight.img_winCenter = msgsgRight.low;
                msgsgRight.img_winWidth = msgsgRight.high;
        }
    }
}
}

```

```

    }
}
else if(msgsRight.img_visual_type == VISUAL_COLOR)
{
    switch(msgsRight.img_select)
    {
        case RIGHT_IMG_WHOLE:
            msgsRight.img_winCenter = msgsRight.lowColorWhole;
            msgsRight.img_winWidth = msgsRight.highColorWhole;
            break;
        case RIGHT_IMG_ROI:
            msgsRight.img_winCenter = msgsRight.lowColorROI;
            msgsRight.img_winWidth = msgsRight.highColorROI;
            break;
        case RIGHT_IMG_REF:
            msgsRight.img_winCenter = msgsRight.lowColorRef;
            msgsRight.img_winWidth = msgsRight.highColorRef;
            printf(" \n\n      set_Rlowhigh:: Vel REF\n");
            break;
        case RIGHT_IMG_OTHER:
            msgsRight.img_winCenter = msgsRight.lowColorOther;
            msgsRight.img_winWidth = msgsRight.highColorOther;
            break;
        default:
            msgsRight.img_winCenter = msgsRight.low;
            msgsRight.img_winWidth = msgsRight.high;
            break;
    }
}
}
else
{
    if(msgsRight.img_visual_type == VISUAL_GRAY)
    {
        switch(msgsRight.img_select)
        {
            case RIGHT_IMG_WHOLE:
                msgsRight.img_winCenter = msgsRight.lowGrayWhole;
                msgsRight.img_winWidth = msgsRight.highGrayWhole;
                break;
            case RIGHT_IMG_ROI:
                msgsRight.img_winCenter = msgsRight.lowGrayROI;
                msgsRight.img_winWidth = msgsRight.highGrayROI;
                break;
            case RIGHT_IMG_REF:
                msgsRight.img_winCenter = msgsRight.lowGrayRef;
                msgsRight.img_winWidth = msgsRight.highGrayRef;
                printf(" \n\n      set_Rlowhigh:: REF\n");
                break;
            case RIGHT_IMG_OTHER:
                msgsRight.img_winCenter = msgsRight.lowGrayOther;
                msgsRight.img_winWidth = msgsRight.highGrayOther;
                break;
            default:
                msgsRight.img_winCenter = msgsRight.low;
                msgsRight.img_winWidth = msgsRight.high;
                break;
        }
    }
}
else if(msgsRight.img_visual_type == VISUAL_COLOR)
{
    switch(msgsRight.img_select)
    {
        case RIGHT_IMG_WHOLE:

```

```

        msgsRight.img_winCenter = msgsRight.lowColorWhole;
        msgsRight.img_winWidth = msgsRight.highColorWhole;
        break;
    case RIGHT_IMG_ROI:
        msgsRight.img_winCenter = msgsRight.lowColorROI;
        msgsRight.img_winWidth = msgsRight.highColorROI;
        break;
    case RIGHT_IMG_REF:
        msgsRight.img_winCenter = msgsRight.lowColorRef;
        msgsRight.img_winWidth = msgsRight.highColorRef;
        printf(" \n\n      set_Rlowhigh:: REF\n");
        break;
    case RIGHT_IMG_OTHER:
        msgsRight.img_winCenter = msgsRight.lowColorOther;
        msgsRight.img_winWidth = msgsRight.highColorOther;
        break;
    default:
        msgsRight.img_winCenter = msgsRight.low;
        msgsRight.img_winWidth = msgsRight.high;
        break;
    }
}
}

//
// Update ImgGE::_img
//
//      Given:   img_number
//
//      Find:
//
//      (1)  ImgGE::imgdata (short **, 16 bit)
//           (The original image obtained from files "fname")
//

ImgGE *ObjectManager::get_ImgGE(int img_number, int img_type, int img_pcmra_type, ImgGE
{
    char  fname[300];
    char  tmpStr[30];
    int   tmp;
    ImgGE *magGE = NULL;

    //printf("      get_ImgGE:: %d %d %d \n", img_number, img_type, img_pcmra_type);

    if(img_type == IMAGE_PCMRA)
    {
        if(img_pcmra_type == PCMRA_MAGNITUDE)
            tmp = msgsLoaded.img_start2 + (img_number - msgsLoaded.img_start);
        else if(img_pcmra_type == PCMRA_PHASE)
            tmp = img_number;
        else if(img_pcmra_type == PCMRA_VELOCITY)
        {
            tmp = msgsLoaded.img_start2 + (img_number - msgsLoaded.img_start);
            printf(" %d %d %d %d\n", img_number, msgsLoaded.img_start,
                msgsLoaded.img_start2, tmp);

            sprintf(fname, "%s/E%dS%dI%d.MR", msgsLoaded.img_dir,
                msgsLoaded.img_exam, msgsLoaded.img_series, tmp);
            magGE = new ImgGE(fname);
            tmp = img_number;
        }
        sprintf(tmpStr, "MR");
    }
    else
    {

```

```

    tmp = img_number;
    sprintf(tmpStr, "%d", msgsLoaded.img_type);
}

```

105

```

sprintf(fname, "%s/E%dS%dI%d.%s", msgsLoaded.img_dir, msgsLoaded.img_exam,
        msgsLoaded.img_series, tmp, tmpStr);

```

```

printf("    %s\n", fname);
if(imgGE == NULL) imgGE = new ImgGE(fname);
else imgGE -> set(fname);

```

```

if(img_type == IMAGE_PCMRA &&
    img_pcmra_type == PCMRA_VELOCITY)
{

```

```

    //if(_magImg != NULL) delete _magImg;
    //if(_phaImg != NULL) delete _magImg;
    //_magImg = magGE;
    //_phaImg = imgGE -> copy();

```

```

    Utility *u = new Utility();
    short **img = u -> ToVelocity(imgGE->get_header(), imgGE->get_width(),
    imgGE->get_height(), magGE->get_imgdata(), imgGE->get_imgdata(),
    msgsLeft.posThresh, msgsLeft.negThresh, msgsLeft.magThresh);
    imgGE -> set_imgdata(img);

```

```

    delete u;
    delete magGE;
}

```

```

return imgGE;
}

```

```

ImgGE *ObjectManager::get_ImgGE2(int img_number, ImgGE *imgGE)
{

```

```

    Utility *u = new Utility();

```

```

    if(msgsRight.img_select == RIGHT_IMG_WHOLE)
    {

```

```

        msgsRight.img_type = msgsLeft.img_type;
        msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;
        msgsRight.img_zoom = msgsLeft.img_zoom;

```

```

        if(imgGE == NULL) imgGE = new ImgGE();
        imgGE -> set(_img -> copy());
    }

```

```

    else if(msgsRight.img_select == RIGHT_IMG_ROI)
    {

```

```

        msgsRight.img_type = msgsLeft.img_type;
        msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;

```

```

        if(_imgView -> _ROI != NULL && _imgView -> _ROI -> _draw_status)
        {

```

```

            _imgView -> _ROI -> get_BoundingBox(&msgsLeft.roi_x, &msgsLeft.roi_y,
            &msgsLeft.roi_w, &msgsLeft.roi_h );

```

```

            int x1 = int( float(msgsLeft.roi_x)/_imgView->_zoom );
            int y1 = int( float(msgsLeft.roi_y)/_imgView->_zoom );
            int w1 = int( float(msgsLeft.roi_w)/_imgView->_zoom );
            int h1 = int( float(msgsLeft.roi_h)/_imgView->_zoom );

```

```

            printf(" \n\n get_GE2 ROI:  %d %d  %d %d\n", x1, y1, w1, h1);

```

```

            if(msgsRight.img_type == IMAGE_PCMRA &&
                msgsRight.img_pcmra_type == PCMRA_VELOCITY)
            {

```

```

                if(msgsRight.velocity_select == VELOCITY_ASIS)

```

```

{
    imgGE -> set(x1, y1, w1, h1, _img -> get_imgdata());
}
else if(msgsRight.velocity_select == VELOCITY_AUTO)
{
    /*
    unsigned char **mask = _magImg -> thresh(x1, y1, w1, h1, msgsRight.low_magt

    Utility *u = new Utility();
    short **img = u -> ToVelocityROI(_phaImg->get_header(),
    _magImg->get_imgdata(), _phaImg->get_imgdata(),
    x1, y1, w1, h1, mask, msgsLeft.postThresh,
    msgsLeft.negThresh, msgsLeft.magThresh);

    imgGE -> set_imgdata(img);
    delete u;
    */
}
else if(msgsRight.velocity_select == VELOCITY_ROIMASKED ||
        msgsRight.velocity_select == VELOCITY_FLOWMASKED)
{
    if(_imgView2 -> _ROI != NULL && _imgView2 -> _ROI -> _areaOrg != NULL)
    {
        imgGE -> set(x1, y1, w1, h1, _img -> get_imgdata(), _imgView2 -> _ROI -
    }
    else
        imgGE -> set(x1, y1, w1, h1, _img -> get_imgdata());
}
}
else
    imgGE -> set(x1, y1, w1, h1, _img -> get_imgdata());

msgsRight.img_zoom = get_Rzoom(imgGE->get_width(), imgGE->get_height());
printf("    get_GE2 ROI:    %f\n", msgsRight.img_zoom);

}
else
{
    msgsRight.img_zoom = msgsLeft.img_zoom;
    if(imgGE == NULL) imgGE = new ImgGE();
    imgGE -> set(_img -> copy());
}
}
else if(msgsRight.img_select == RIGHT_IMG_REF)
{
    char  fname[300];

    msgsRight.img_zoom = msgsLeft.img_zoom;

    sprintf(fname, "%s/%s", msgsLoaded.img_dir, msgsLoaded.img_ref);
    if(imgGE == NULL) imgGE = new ImgGE(fname);
    else imgGE -> set(fname);
}
else if(msgsRight.img_select == RIGHT_IMG_OTHER)
{
    msgsRight.img_type = msgsLeft.img_type;
    msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;
    msgsRight.img_zoom = msgsLeft.img_zoom;

    if(imgGE == NULL) imgGE = new ImgGE();
    imgGE -> set(_img -> copy());
}

}

delete u;
return imgGE;

```

```

}

float ObjectManager::get_RZoom(int w, int h)
{
    int tmp;
    if(w > h) tmp = int(512.0/float(w));
    else tmp = int(512.0/float(h));
    return float(tmp);
}

void ObjectManager::show2D()
{
    if(_imgView2 != NULL && !msgsRight.show_status)
    {
        _imgView2 -> show();
        msgsRight.show_status = TRUE;
    }
}

void ObjectManager::hide2D()
{
    if(_imgView2 != NULL && msgsRight.show_status)
    {
        _imgView2 -> hide();
        msgsRight.show_status = FALSE;
    }
}

void ObjectManager::showL2D()
{
    if(_imgView != NULL && !msgsLeft.show_status)
    {
        _imgView -> show();
        msgsLeft.show_status = TRUE;
    }
}

void ObjectManager::hideL2D()
{
    if(_imgView != NULL && msgsLeft.show_status)
    {
        _imgView -> hide();
        msgsLeft.show_status = FALSE;
    }
}

////////////////////////////////////
//
// 3D Stuff
//
////////////////////////////////////

void ObjectManager::remove_animate3D()
{
    if(_animate->_soiv != NULL)
    {
        for(int i=0; i<_animate->_num_imgs; i++)
            if(_animate->_soiv[i]._iv != NULL)
            {
                _animate->_soiv[i]._iv -> unref();
                _animate->_soiv[i]._iv = NULL;
            }
        delete _animate->_soiv;
        _animate->_soiv = NULL;
    }
}

```

```

}

void ObjectManager::create_animate3D()
{
    printf("    START    create_animate  3D\n");
    int i, j, i1, j1;

    if(_R3D == NULL || msgsRight.num_imgs < 2)
    {
        _animate->_ivview = NULL;
        _animate->_soiv = NULL;
        return;
    }

    if( _animate->_widget == NULL )
    {
        _animate->_num_imgs = msgsRight.num_imgs;
        _animate->_widget = _R3D -> baseWidget();
    }

    _animate->_soiv = new SOIV[msgsRight.num_imgs];
    for(i=0; i<msgsRight.num_imgs; i++)
        _animate->_soiv[i]._iv = NULL;

    _animate->_ivview = _R3D;

    msgsRight.img_type = msgsLeft.img_type;
    msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;

    update_progress("Create 3D ScenGraphs For Animation");

    Progress_Animate3D();
}

```

```

void ObjectManager::create_animateSymphony()
{
    int i;

    if( _animate->_widget == NULL )
    {
        _animate->_num_imgs = msgsRight.num_imgs;
        _animate->_widget = _imgView2 -> baseWidget();
    }

    _animate->_pixmap = new Pixmap[msgsRight.num_imgs];
    for(i=0; i<msgsRight.num_imgs; i++)
        _animate->_pixmap[i] = NULL;

    _animate->_width = _imgView2 -> get_width();
    _animate->_height = _imgView2 -> get_height();

    _animate -> _gc = DefaultGCOFScreen(XtScreen(_animate->_widget));

    msgsRight.img_type = msgsLeft.img_type;
    msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;

    if(_L3D == NULL || msgsRight.num_imgs < 2)
    {
        _animate->_ivview = NULL;
        _animate->_soiv = NULL;
    }
    else
    {
        _animate->_ivview = _L3D;
        _animate->_soiv = new SOIV[msgsRight.num_imgs];
        for(i=0; i<msgsRight.num_imgs; i++)

```

```

    _animate->_soiv[i] iv = NULL;
}

update_progress("Create 2D & 3D Images For Animation");

Progress_AnimateSymphony();
}

void ObjectManager::empty_animate3D()
{
    _animate-> _soiv = NULL;
    _animate-> _ivview = NULL;
}

void ObjectManager::hide3D()
{
    if(_R3D != NULL) _R3D -> hide();
}

void ObjectManager::hideL3D()
{
    if(_L3D != NULL) _L3D -> hide();
}

void ObjectManager::update_RimgView3D()
{
    hide2D();

    Utility_3D *u3D = new Utility_3D();

    int w = _img2->get_width();
    int h = _img2->get_height();
    short **img = _img2->get_imgdata();

    //int w = _imgView2->get_width();
    //int h = _imgView2->get_height();
    //short **img = _imgView2->_zoomImg;

    u3D -> to_ivFile(w, h, img, &(msgsRight.ratio3D), msgsRight.camera,
        &(msgsRight.Height3D), &(msgsRight.YPos3D), msgsRight.Fixed3D,
        msgsRight.flow3DDir);

    _R3D = u3D -> create_iv("tmp.iv", _bb->baseWidget(),
        _R3D, 670, 90, 500, 500);

    /*if(msgsRight.publish == PUBLISH_3DFLOW)
        ((BbFormat *)_RFormat) -> savePublish(PUBLISH_3DFLOW);
    */
    delete u3D;
}

void ObjectManager::update_RimgView3DROI()
{
    hide2D();

    Utility_3D *u3D = new Utility_3D();

    _R3D = u3D -> create_iv("ROIS.iv", _bb->baseWidget(),
        _R3D, 670, 90, 500, 500);

    delete u3D;
}

void ObjectManager::update_LimgView3D()
{

```



```

hideL2D();

Utility_3D *u3D = new Utility_3D();

//int w = _imgView2->get_width();
//int h = _imgView2->get_height();
//short **img = _imgView2->_zoomImg;
int w = _img2->get_width();
int h = _img2->get_height();
short **img = _img2->get_imgdata();

u3D -> to_ivFile(w, h, img, &(msgsRight.ratio3D), msgsRight.camera,
    &(msgsRight.Height3D), &(msgsRight.YPos3D), msgsRight.Fixed3D,
    msgsRight.flow3DDir);

_L3D = u3D -> create_iv("tmp.iv", _bb->baseWidget(),
    _L3D, 50, 90, 500, 500);

delete u3D;
}

void ObjectManager::update_win3D()
{
    if(_win3D == NULL)
    {
        _win3D = new Win3DMainWindow("win3D");
        _win3D -> show();
        XResizeWindow(XtDisplay(_win3D->baseWidget()), XtWindow(_win3D->baseWidget()),
            600, 500);
        ((Win3DMainWindow *)_win3D) -> set(this);
        ((Win3DMainWindow *)_win3D) -> update();
    }
    else
    {
        ((Win3DMainWindow *)_win3D) -> update();
    }
}

//---- End editable code block: End of generated code

```

```
#include "ReadConfig.h"
```

111

```
#include "Utility.h"
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
MessagesLoaded ReadConfig()
```

```
{
    MessagesLoaded msgs;
    FILE *fp;

    fp = fopen("input.dat", "r");
    fscanf(fp, "%s", msgs.img_dir);
    fscanf(fp, "%s", msgs.img_type);
    fscanf(fp, "%s", msgs.img_anatomy);
    fscanf(fp, "%d", &msgs.img_exam);
    fscanf(fp, "%d", &msgs.img_series);
    if(strcmp(msgs.img_type, "PCMRA") == 0)
    {
        fscanf(fp, "%d", &msgs.img_start);
        fscanf(fp, "%d", &msgs.img_end);
        fscanf(fp, "%d", &msgs.img_start2);
        fscanf(fp, "%d", &msgs.img_end2);

        printf(" %d %d %d %d\n", msgs.img_start, msgs.img_end,
            msgs.img_start2, msgs.img_end2);
    }
    else
    {
        fscanf(fp, "%d", &msgs.img_start);
        fscanf(fp, "%d", &msgs.img_end);

        printf(" %d %d\n", msgs.img_start, msgs.img_end);
    }

    fscanf(fp, "%s", msgs.img_ref);

    fclose(fp);

    return msgs;
}
```

```
MessagesLeft ReadConfigLeft(int img_type)
```

```
{
    MessagesLeft msgs;
    FILE *fp;

    fp = fopen("input.Left", "r");

    fscanf(fp, "%f", &msgs.low);
    fscanf(fp, "%f", &msgs.high);

    if(img_type == IMAGE_PCMRA)
    {
        fscanf(fp, "%f", &msgs.lowMag);
        fscanf(fp, "%f", &msgs.highMag);
        fscanf(fp, "%f", &msgs.lowPha);
        fscanf(fp, "%f", &msgs.highPha);
    }

    fclose(fp);

    return msgs;
}
```

```

{
    MessagesRight msgs;
    FILE *fp;

    fp = fopen("input.Right", "r");

    fscanf(fp, "%f", &msgs.lowGrayWhole);
    fscanf(fp, "%f", &msgs.highGrayWhole);
    fscanf(fp, "%f", &msgs.lowGrayROI);
    fscanf(fp, "%f", &msgs.highGrayROI);
    fscanf(fp, "%f", &msgs.lowGrayRef);
    fscanf(fp, "%f", &msgs.highGrayRef);
    fscanf(fp, "%f", &msgs.lowGrayOther);
    fscanf(fp, "%f", &msgs.highGrayOther);

    fscanf(fp, "%f", &msgs.lowColorWhole);
    fscanf(fp, "%f", &msgs.highColorWhole);
    fscanf(fp, "%f", &msgs.lowColorROI);
    fscanf(fp, "%f", &msgs.highColorROI);
    fscanf(fp, "%f", &msgs.lowColorRef);
    fscanf(fp, "%f", &msgs.highColorRef);
    fscanf(fp, "%f", &msgs.lowColorOther);
    fscanf(fp, "%f", &msgs.highColorOther);

    if(img_type == IMAGE_PCMRA)
    {
        fscanf(fp, "%f", &msgs.lowMagGrayWhole);
        fscanf(fp, "%f", &msgs.highMagGrayWhole);
        fscanf(fp, "%f", &msgs.lowMagGrayROI);
        fscanf(fp, "%f", &msgs.highMagGrayROI);
        fscanf(fp, "%f", &msgs.lowMagGrayRef);
        fscanf(fp, "%f", &msgs.highMagGrayRef);
        fscanf(fp, "%f", &msgs.lowMagGrayOther);
        fscanf(fp, "%f", &msgs.highMagGrayOther);

        fscanf(fp, "%f", &msgs.lowMagColorWhole);
        fscanf(fp, "%f", &msgs.highMagColorWhole);
        fscanf(fp, "%f", &msgs.lowMagColorROI);
        fscanf(fp, "%f", &msgs.highMagColorROI);
        fscanf(fp, "%f", &msgs.lowMagColorRef);
        fscanf(fp, "%f", &msgs.highMagColorRef);
        fscanf(fp, "%f", &msgs.lowMagColorOther);
        fscanf(fp, "%f", &msgs.highMagColorOther);

        fscanf(fp, "%f", &msgs.lowPhaGrayWhole);
        fscanf(fp, "%f", &msgs.highPhaGrayWhole);
        fscanf(fp, "%f", &msgs.lowPhaGrayROI);
        fscanf(fp, "%f", &msgs.highPhaGrayROI);
        fscanf(fp, "%f", &msgs.lowPhaGrayRef);
        fscanf(fp, "%f", &msgs.highPhaGrayRef);
        fscanf(fp, "%f", &msgs.lowPhaGrayOther);
        fscanf(fp, "%f", &msgs.highPhaGrayOther);

        fscanf(fp, "%f", &msgs.lowPhaColorWhole);
        fscanf(fp, "%f", &msgs.highPhaColorWhole);
        fscanf(fp, "%f", &msgs.lowPhaColorROI);
        fscanf(fp, "%f", &msgs.highPhaColorROI);
        fscanf(fp, "%f", &msgs.lowPhaColorRef);
        fscanf(fp, "%f", &msgs.highPhaColorRef);
        fscanf(fp, "%f", &msgs.lowPhaColorOther);
        fscanf(fp, "%f", &msgs.highPhaColorOther);
    }
}

```

```
}  
fclose(fp);  
return msgs;  
}
```

```
#include <math.h>
#include <stdio.h>
#include <malloc.h>
#include "Utility_Math.h"
#include "Utility_Widget.h"
```

```
Ellipse::Ellipse(Widget w, int color) : ROI(w, color)
{
}
```

```
Ellipse::~~Ellipse()
{
}
```

```
void Ellipse::init(int x, int y)
{
    _start.x = x;
    _start.y = y;
}
```

```
void Ellipse::motion(int x, int y)
{
    int x0, y0;

    if(_show_status) draw_img();

    _xlen = (int)(fabsf(x - _start.x + 1)/2.0);
    _ylen = (int)(fabsf(y - _start.y + 1)/2.0);

    if (x < _start.x) x0 = x;
    else x0 = _start.x;

    if (y < _start.y) y0 = y;
    else y0 = _start.y;

    _angle = 0.0;

    if(_xlen > 2 && _ylen > 2)
    {
        _points_in_border._numPoints = get_points_in_border();

        if(_points_in_border._numPoints > 0)
        {
            _center.x = x0 + _xlen;
            _center.y = y0 + _ylen;
            _points_in_border.translation1(_center.x, _center.y);
            draw();
            _show_status = TRUE;
        }
        else
        {
            _show_status = FALSE;
        }
    }
    else _show_status = FALSE;
}
```

```
void Ellipse::new_started(int x, int y)
{
    if(!_draw_status) motion(x, y);
}
```

```

void Ellipse::released(int x, int y)
{
    if(_show_status) _draw_status = TRUE;
}

void Ellipse::draw()
{
    _points_in_border.draw(_widget, _gc);
}

void Ellipse::transform(float x1, float y1, float *x, float *y)
{
    *x = (float)x1*cos(_angle) - (float)y1*sin(_angle);
    *y = (float)x1*sin(_angle) + (float)y1*cos(_angle);
}

int Ellipse::get_points_in_border()
{
    if(_xlen < 2 || _ylen < 2) return 0;

    float a = (float) _xlen;
    float b = (float) _ylen;

    float x, y, x1, y1, tmp;
    int i, ip;

    ip = 0;
    Utility_Math *u = new Utility_Math();

    if(_xlen > _ylen)
    {
        for(i=-_xlen; i<=_xlen; i++)
        {
            x1 = (float)i;
            tmp = a*a - (float)(i*i);

            y1 = - fsqrt( tmp ) * b / a;
            transform(x1, y1, &x, &y);

            _points_in_border._points[ip].x = u->int_t(x);
            _points_in_border._points[ip].y = u->int_t(y);

            ip++;
        }
        for(i=_xlen-1; i>=_xlen; i--)
        {
            x1 = (float)i;
            tmp = a*a - (float)(i*i);

            y1 = fsqrt( tmp ) * b / a;
            transform(x1, y1, &x, &y);

            _points_in_border._points[ip].x = u->int_t(x);
            _points_in_border._points[ip].y = u->int_t(y);

            ++ip;
        }
    }
    else
    {
        for(i=-_ylen; i<=_ylen; i++)
        {
            y1 = (float)i;

```

```

        tmp = b*b-(float)(i*i);

        x1 = -fsqrt(tmp) * a / b;
        transform(x1, y1, &x, &y);

        _points_in_border._points[ip].x = u->int_t(x);
        _points_in_border._points[ip].y = u->int_t(y);

        ++ip;
    }
    for(i=_ylen-1; i>=_ylen; i--)
    {
        y1 = (float)i;
        tmp = b*b-(float)(i*i);

        x1 = fsqrt(tmp) * a / b;
        transform(x1, y1, &x, &y);

        _points_in_border._points[ip].x = u->int_t(x);
        _points_in_border._points[ip].y = u->int_t(y);

        ++ip;
    }
}

delete u;
return ip;
}

void Ellipse::init_move(int x, int y)
{
    _start.x = x;
    _start.y = y;
}

void Ellipse::init_modify(int x, int y)
{
    _start.x = x;
    _start.y = y;

    if(x > _center.x && y > _center.y)
        _corner = RIGHT_BOTTOM;
    else if(x > _center.x && y <= _center.y)
        _corner = RIGHT_TOP;
    if(x <= _center.x && y > _center.y)
        _corner = LEFT_BOTTOM;
    else if(x <= _center.x && y <= _center.y)
        _corner = LEFT_TOP;
}

void Ellipse::motion_move(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;

    int x1, y1;
    int w1;
    int h1;

    int x0 = _center.x - _xlen;
    int y0 = _center.y - _ylen;
    int w0 = 2*_xlen;
    int h0 = 2*_ylen;

    x1 = x0 + dx;
    y1 = y0 + dy;

```

```

if(x1 < 0) x1 = 0;
else if(x1 > (_width - 1 - w0)) x1 = _width - 1 - w0;
if(y1 < 0) y1 = 0;
else if(y1 > (_height - 1 - h0)) y1 = _height - 1 - h0;

if(_show_status) draw_img();
else _show_status = TRUE;

_center.x = x1 + _xlen;
_center.y = y1 + _ylen;

_start.x = x;
_start.y = y;

_points_in_border._numPoints = get_points_in_border();
if(_points_in_border._numPoints > 0)
    _points_in_border.translation1(_center.x, _center.y);
draw();
}

```

```

void Ellipse::motion_modify(int x, int y)
{

```

```

    int dx = x - _start.x;
    int dy = y - _start.y;

```

```

    int x1, y1, w1, h1;
    int xlen, ylen, xflag, yflag;

```

```

    int x0 = int(_center.x) - _xlen;
    int y0 = int(_center.y) - _ylen;
    int w0 = 2*_xlen;
    int h0 = 2*_ylen;

```

```

    switch (_corner)
    {

```

```

        case LEFT_TOP:

```

```

            x1 = x0 + dx;
            y1 = y0 + dy;
            w1 = w0 - dx;
            h1 = h0 - dy;
            xflag = 1;
            yflag = 1;
            break;

```

```

        case LEFT_BOTTOM:

```

```

            x1 = x0 + dx;
            y1 = y0;
            w1 = w0 - dx;
            h1 = h0 + dy;
            xflag = 1;
            yflag = -1;
            break;

```

```

        case RIGHT_TOP:

```

```

            x1 = x0;
            y1 = y0 + dy;
            w1 = w0 + dx;
            h1 = h0 - dy;
            xflag = -1;
            yflag = 1;
            break;

```

```

        case RIGHT_BOTTOM:

```

```

            x1 = x0;
            y1 = y0;
            w1 = w0 + dx;
            h1 = h0 + dy;
            xflag = -1;

```



```

        yflag = -1;
        break;
    default:
        break;
}

xlen = w1/2;
ylen = h1/2;

if( w1 > 2 && h1 > 2 && (xlen != _xlen || ylen != _ylen) )
{
    if(_show_status) draw_img();
    else _show_status = TRUE;

    _center.x += xflag*(_xlen - xlen);
    _center.y += yflag*(_ylen - ylen);

    _xlen = xlen;
    _ylen = ylen;

    _start.x = x;
    _start.y = y;

    _points_in_border._numPoints = get_points_in_border();
    if(_points_in_border._numPoints > 0)
        _points_in_border.translation1(_center.x, _center.y);
    draw();
}

}

void Ellipse::fill()
{
    Utility_Widget *u = new Utility_Widget();
    GC gc = u -> get_GC(_widget);

    XPoint *xpoint = _points_in_border.get_XPoint();

    XFillPolygon(XtDisplay(_widget), XtWindow(_widget), gc, xpoint,
        _points_in_border._numPoints, Complex, CoordModeOrigin);

    XtReleaseGC(_widget, gc);
    delete xpoint;
    delete u;
}

void Ellipse::get_BoundingBox(int *x, int *y, int *w, int *h)
{
    float minX, maxX, minY, maxY;

    _points_in_border.get_minmax(&minX, &maxX, &minY, &maxY);
    *x = int(minX);
    *y = int(minY);
    *w = int(maxX - minX);
    *h = int(maxY - minY);
}

```

#include <math.h>

#include <stdio.h>

#include "Utility_Widget.h"

```
Rectangle::Rectangle(Widget w, int color) : ROI(w, color)
{
}
```

```
Rectangle::~Rectangle()
{
}
```

```
void Rectangle::draw()
{
    Utility_Widget *u = new Utility_Widget();
    //get_points_in_border();
    u -> draw_rectangle(_widget, _gc, _x, _y, _w, _h);
    delete u;
}
```

```
void Rectangle::get_points_in_border()
{
    _points_in_border._points[0].x = _x;
    _points_in_border._points[0].y = _y;
    _points_in_border._points[1].x = _x + _w;
    _points_in_border._points[1].y = _y;
    _points_in_border._points[2].x = _x + _w;
    _points_in_border._points[2].y = _y + _h;
    _points_in_border._points[3].x = _x;
    _points_in_border._points[3].y = _y + _h;
    _points_in_border._numPoints = 4;
}
```

```
void Rectangle::fill()
{
    Utility_Widget *u = new Utility_Widget();

    get_points_in_border();
    XPoint *xpoint = _points_in_border.get_XPoint();

    XFillPolygon(XtDisplay(_widget), XtWindow(_widget), _gc, xpoint,
        _points_in_border._numPoints, Complex, CoordModeOrigin);

    delete xpoint;
    delete u;
}
```

```
void Rectangle::get_BoundingBox(int *x, int *y, int *w, int *h)
{
    *x = _x;
    *y = _y;
    *w = _w;
    *h = _h;
}
```

```
void Rectangle::init(int x, int y)
{
    _start.x = x;
    _start.y = y;
}
```

```
void Rectangle::new_started(int x, int y)
{
}
```

```

    if(!_draw_status) motion(x, y);
}

void Rectangle::released(int x, int y)
{
    if(_show_status)
    {
        _center.x = _x + _w/2;
        _center.y = _y + _h/2;
        _draw_status = TRUE;
    }
}

void Rectangle::motion(int x, int y)
{
    if(_show_status) draw_img();

    if (x < _start.x) _x = x;
    else _x = _start.x;

    if (y < _start.y) _y = y;
    else _y = _start.y;

    _w = (int)(fabsf(x - _start.x + 1));
    _h = (int)(fabsf(y - _start.y + 1));

    if(_w > 2 && _h > 2)
    {
        draw();
        _show_status = TRUE;
    }
    else _show_status = FALSE;
}

void Rectangle::init_move(int x, int y)
{
    _start.x = x;
    _start.y = y;
}

void Rectangle::init_modify(int x, int y)
{
    _start.x = x;
    _start.y = y;

    if(x > _center.x && y > _center.y)
        _corner = RIGHT_BOTTOM;
    else if(x > _center.x && y <= _center.y)
        _corner = RIGHT_TOP;
    if(x <= _center.x && y > _center.y)
        _corner = LEFT_BOTTOM;
    else if(x <= _center.x && y <= _center.y)
        _corner = LEFT_TOP;
}

void Rectangle::motion_move(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;
    int x1, y1;

    x1 = _x + dx;
    y1 = _y + dy;

    if(x1 < 0) x1 = 0;
    else if(x1 > (_width - 1 - _w)) x1 = _width - 1 - _w;

```

```

    if(y1 < 0) y1 = 0;
    else if(y1 > (_height - 1 - _h)) y1 = _height - 1 - _h;

    if(_show_status) draw_img();
    else _show_status = TRUE;

    _x = x1;
    _y = y1;

    _start.x = x;
    _start.y = y;

    draw();
}

void Rectangle::motion_modify(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;
    int x1, y1;

    int w1;
    int h1;

    switch (_corner)
    {
        case LEFT_TOP:
            x1 = _x + dx;
            y1 = _y + dy;
            w1 = _w - dx;
            h1 = _h - dy;
            break;
        case LEFT_BOTTOM:
            x1 = _x + dx;
            y1 = _y;
            w1 = _w - dx;
            h1 = _h + dy;
            break;
        case RIGHT_TOP:
            x1 = _x;
            y1 = _y + dy;
            w1 = _w + dx;
            h1 = _h - dy;
            break;
        case RIGHT_BOTTOM:
            x1 = _x;
            y1 = _y;
            w1 = _w + dx;
            h1 = _h + dy;
            break;
        default:
            break;
    }

    if(w1 > 2 && h1 > 2)
    {
        if(_show_status) draw_img();
        else _show_status = TRUE;

        _x = x1;
        _y = y1;
        _w = w1;
        _h = h1;

        _start.x = x;
        _start.y = y;
    }
}

```

```
_center.x = _x + _r;  
_center.y = _y + _r/2;
```

122

```
draw();
```

```
}
```

```
}
```

User: meide
Host: phoenix
Class: phoenix
Job: ImgAlloc.C

123

```
#include <math.h>
#include <stdio.h>
#include "Utility_Math.h"
#include "Utility_Widget.h"
#include "Utility_Vision.h"
```

```
Polygon::Polygon(Widget w, int color) : ROI(w, color)
{
    _modify_num = -1;
}
```

```
Polygon::~~Polygon()
{
}
```

```
void Polygon::draw()
{
    _points_in_border.draw(_widget, _gc);
}
```

```
void Polygon::init(int x, int y)
{
    _start.x = x;
    _start.y = y;
    _points_in_border.add(_start.x, _start.y);
}
```

```
void Polygon::motion(int x, int y)
{
    Utility_Widget *uw = new Utility_Widget();
    Utility_Math *um = new Utility_Math();

    if(_show_status)
    {
        _points_in_border.add(_x, _y);
        draw_img();
        --(_points_in_border._numPoints);
        _points_in_border.draw_noloop(_widget, _gc);
    }

    _x = x;
    _y = y;

    if( um->distance(_start.x, _start.y, _x, _y) > 2)
    {
        uw -> draw_line(_widget, _gc, _x, _y, _start.x, _start.y);
        _show_status = TRUE;
    }
    else
    {
        _show_status = FALSE;
    }

    delete um;
    delete uw;
}
```

```
void Polygon::released(int x, int y)
{
    if( _show_status )
    {
        _points_in_border.add(x, y);
    }
}
```

```

    _start.x = x;
    _start.y = y;
    _show_status = FALSE;
}
}

void Polygon::new_started(int x, int y)
{
    motion(x, y);
}

void Polygon::finished(int x, int y)
{
    _x = _points_in_border._points[0].x;
    _y = _points_in_border._points[0].y;

    Utility_Widget *u = new Utility_Widget();
    u -> draw_line(_widget, _gc, _x, _y, _start.x, _start.y);
    _show_status = TRUE;
    _draw_status = TRUE;
    delete u;
    AcceptROI();
}

void Polygon::init_move(int x, int y)
{
    _start.x = x;
    _start.y = y;
}

void Polygon::init_modify(int x, int y)
{
    _start.x = x;
    _start.y = y;

    _modify_num = _points_in_border.closest(x, y);
}

void Polygon::released_modify(int x, int y)
{
}

void Polygon::motion_move(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;

    int x1, y1, w1, h1;

    float x0f, y0f, x2f, y2f;
    _points_in_border.get_minmax(&x0f, &x2f, &y0f, &y2f);

    int x0 = int(x0f);
    int y0 = int(y0f);
    int w0 = int(x2f - x0f);
    int h0 = int(y2f - y0f);

    if(_show_status) draw_img();
    else _show_status = TRUE;

    x1 = x0 + dx;
    y1 = y0 + dy;

    if(x1 < 0) x1 = 0;
    else if(x1 > (_width - 1 - w0)) x1 = _width - 1 - w0;

```



```

    if(y1 < 0) y1 = 0;
    else if(y1 > (_height - 1 - h0)) y1 = _height - 1 - h0;

    dx = x1 - x0;
    dy = y1 - y0;
    _points_in_border.translation2(dx, dy);

    _start.x = x;
    _start.y = y;
    draw();
}

```

```

void Polygon::motion_modify(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;

    int x1, y1, w1, h1;

    float x0f, y0f, x2f, y2f;
    _points_in_border.get_minmax(&x0f, &x2f, &y0f, &y2f);

    int x0 = int(x0f);
    int y0 = int(y0f);
    int w0 = int(x2f - x0f);
    int h0 = int(y2f - y0f);

    if(_show_status) draw_img();
    else _show_status = TRUE;
}

```

```

if(_modify_num >= 0)
{
    _points_in_border._points[_modify_num].x += dx;
    _points_in_border._points[_modify_num].y += dy;
}

_start.x = x;
_start.y = y;
draw();
}

```

```

void Polygon::fill()
{
    Utility_Widget *u = new Utility_Widget();
    GC gc = u -> get_GC(_widget);

    XPoint *xpoint = _points_in_border.get_XPoint();

    XFillPolygon(XtDisplay(_widget), XtWindow(_widget), gc, xpoint,
        _points_in_border._numPoints, Complex, CoordModeOrigin);

    XtReleaseGC(_widget, gc);
    delete xpoint;
    delete u;
}

```

```

void Polygon::get_BoundingBox(int *x, int *y, int *w, int *h)
{
    float minX, maxX, minY, maxY;

    _points_in_border.get_minmax(&minX, &maxX, &minY, &maxY);
    *x = int(minX);
    *y = int(minY);
    *w = int(maxX - minX);
    *h = int(maxY - minY);
}

```


#include <math.h>

#include <stdio.h>

#include "Utility_Widget.h"

#include "Utility_Vision.h"

```
FreeHand::FreeHand(Widget w, int color) : ROI(w, color)
{
}
```

```
FreeHand::~FreeHand()
{
}
```

```
void FreeHand::draw()
{

```

```
    _points_in_border.draw(_widget, _gc);
}
```

```
void FreeHand::init(int x, int y)
{

```

```
    _x = x;
```

```
    _y = y;
```

```
    _points_in_border.init(x, y);
```

```
    _show_status = TRUE;
```

```
    Utility_Widget *u = new Utility_Widget();
```

```
    u -> draw_point(_widget, _gc, _x, _y);
```

```
    delete u;
```

```
}
```

```
void FreeHand::motion(int x, int y)
{

```

```
    if( !_draw_status && (_x != x || _y != y))
```

```
    {
```

```
        _x = x;
```

```
        _y = y;
```

```
        _points_in_border.add(x, y);
```

```
        Utility_Widget *u = new Utility_Widget();
```

```
        // u -> draw_point(_widget, x, y, _width, _height, _cimg);
```

```
        u -> draw_point(_widget, _gc, _x, _y);
```

```
        delete u;
```

```
    }
```

```
}
```

```
void FreeHand::released(int x, int y)
{

```

```
}
```

```
void FreeHand::new_started(int x, int y)
{

```

```
    if( !_draw_status ) motion(x, y);
```

```
}
```

```
void FreeHand::finished(int x, int y)
{

```

```
    if( !_draw_status ) _draw_status = TRUE;
```

```
    AcceptROI();
```

```
}
```

```
void FreeHand::init_move(int x, int y)
{

```

```
    _start.x = x;
```

```

    _start.y = y;
}

void FreeHand::init_modify(int x, int y)
{
    //_start.x = x;
    //_start.y = y;
    _x = x;
    _y = y;
    _modify_num = _points_in_border.closest(x, y);

    printf("    init_modify  %d  %d      %d\n", x, y, _modify_num);

    if(_modify_num != -1) init_modify2(_modify_num);
}

void FreeHand::init_modify2(int num)
{
    int n = NUM_POINTS;
    int k = 0;
    int i;

    _start.x = _points_in_border._points[num].x;
    _start.y = _points_in_border._points[num].y;

    for(i=num+1; i<_points_in_border._numPoints; i++)
    {
        _tmpPN1[k] = i;
        _tmpPoints1[k].x = _points_in_border._points[i].x;
        _tmpPoints1[k].y = _points_in_border._points[i].y;
        ++k;
        if(k == n) break;
    }
    _numPoints1 = k;

    k = 0;
    for(i=num-1; i>=0; i--)
    if(i >= 0)
    {
        _tmpPN2[k] = i;
        _tmpPoints2[k].x = _points_in_border._points[i].x;
        _tmpPoints2[k].y = _points_in_border._points[i].y;
        ++k;
        if(k == n) break;
    }
    _numPoints2 = k;

    if(k < n)
    {
        k = _numPoints2;
        for(i=_points_in_border._numPoints - 1; i>=0; i--)
        {
            if(i == num) break;
            else
            {
                _tmpPN2[k] = i;
                _tmpPoints2[k].x = _points_in_border._points[i].x;
                _tmpPoints2[k].y = _points_in_border._points[i].y;
                ++k;
                if(k == n) break;
            }
        }
        _numPoints2 = k;
    }
}

```

```

void FreeHand::released_modify(int x, int y)
{
    /*
        fill();

        if(_area != NULL)
        {
            Utility_Vision *u = new Utility_Vision();
            u -> freeImg(_area);
            delete u;
        }

        Utility_Widget *u = new Utility_Widget();
        _area = u -> get_mask(_widget, _width, _height);
        delete u;
    */
}

void FreeHand::motion_move(int x, int y)
{
    int dx = x - _start.x;
    int dy = y - _start.y;

    int x1, y1, w1, h1;

    float x0f, y0f, x2f, y2f;
    _points_in_border.get_minmax(&x0f, &x2f, &y0f, &y2f);

    int x0 = int(x0f);
    int y0 = int(y0f);
    int w0 = int(x2f - x0f);
    int h0 = int(y2f - y0f);

    if(_show_status) draw_img();
    else _show_status = TRUE;

    x1 = x0 + dx;
    y1 = y0 + dy;

    if(x1 < 0) x1 = 0;
    else if(x1 > (_width - 1 - w0)) x1 = _width - 1 - w0;
    if(y1 < 0) y1 = 0;
    else if(y1 > (_height - 1 - h0)) y1 = _height - 1 - h0;

    dx = x1 - x0;
    dy = y1 - y0;
    _points_in_border.translation2(dx, dy);

    _start.x = x;
    _start.y = y;
    draw();
}

void FreeHand::motion_modify(int x, int y)
{
    //int dx = x - _start.x;
    //int dy = y - _start.y;
    int dx = x - _x;
    int dy = y - _y;
    /*
        int x1, y1, w1, h1;

        float x0f, y0f, x2f, y2f;
        _points_in_border.get_minmax(&x0f, &x2f, &y0f, &y2f);
    */
}

```

```

int x0 = int(x0f);
int y0 = int(y0f);
int w0 = int(x2f - x0f);
int h0 = int(y2f - y0f);
*/

if(_show_status) draw_img();
else _show_status = TRUE;

if(_modify_num >= 0)
{
    //_points_in_border._points[_modify_num].x += dx;
    //_points_in_border._points[_modify_num].y += dy;
    _points_in_border._points[_modify_num].x = _start.x + dx;
    _points_in_border._points[_modify_num].y = _start.y + dy;

    int i, x2, y2;
    for(i=0; i<_numPoints1; i++)
    {
        x2 = _tmpPoints1[i].x + scaling(dx, i+1);
        if(x2 < 0) x2 = 0;
        _points_in_border._points[_tmpPN1[i]].x = x2;

        y2 = _tmpPoints1[i].y + scaling(dy, i+1);
        if(y2 < 0) y2 = 0;
        _points_in_border._points[_tmpPN1[i]].y = y2;
    }
    for(i=0; i<_numPoints2; i++)
    {
        x2 = _tmpPoints2[i].x + scaling(dx, i+1);
        if(x2 < 0) x2 = 0;
        _points_in_border._points[_tmpPN2[i]].x = x2;

        y2 = _tmpPoints2[i].y + scaling(dy, i+1);
        if(y2 < 0) y2 = 0;
        _points_in_border._points[_tmpPN2[i]].y = y2;
    }
}

//_start.x = x;
//_start.y = y;
draw();
}

int FreeHand::scaling(int diff, int len)
{
    float tmp;
    float a = 1.0;

    tmp = fabsf(diff) - a * float(len);
    if(tmp < 0) tmp = 0;

    if(diff < 0) tmp = - tmp;

    return int(tmp);
}

void FreeHand::fill()
{
    Utility_Widget *u = new Utility_Widget();
    GC gc = u -> get_GC(_widget);

    XPoint *xpoint = _points_in_border.get_XPoint();

    XFillPolygon(XtDisplay(_widget), XtWindow(_widget), gc, xpoint,
        _points_in_border._numPoints, Complex, CoordModeOrigin);
}

```

```
XtReleaseGC(_widget, g);
delete xpoint;
delete u;
}

void FreeHand::get_BoundingBox(int *x, int *y, int *w, int *h)
{
    float minX, maxX, minY, maxY;

    _points_in_border.get_minmax(&minX, &maxX, &minY, &maxY);
    *x = int(minX);
    *y = int(minY);
    *w = int(maxX - minX);
    *h = int(maxY - minY);
}
```

```
#include "Points.h"
```

```
#include <stdio.h>
```

```
#include <math.h>
```

```
#include "Utility_Widget.h"
```

```
#include "Utility_Math.h"
```

```
Points::Points()
```

```
{
    _currPoints = 0;
    _numPoints = 0;
}
```

```
Points::~~Points()
```

```
{
}
```

```
void Points::clear()
```

```
{
    _currPoints = 0;
    _numPoints = 0;
}
```

```
void Points::init(int x, int y)
```

```
{
    _points[0].x = (float)x;
    _points[0].y = (float)y;
    _numPoints = 1;
}
```

```
void Points::add(int x, int y)
```

```
{
    _points[_numPoints].x = (float)x;
    _points[_numPoints].y = (float)y;
    ++_numPoints;
}
```

```
void Points::add(float x, float y)
```

```
{
    _points[_numPoints].x = x;
    _points[_numPoints].y = y;
    ++_numPoints;
}
```

```
void Points::draw_img(Widget w, int w1, int h1, ColorImage *cimg)
```

```
{
    if(_numPoints > 0)
    {
        int i, tmp;
        Point p1, p2;

        Utility_Widget *uw = new Utility_Widget();

        for(i=0; i<(_numPoints-1); i++)
        {
            p1.x = _points[i].x;
            p1.y = _points[i].y;
            p2.x = _points[i+1].x;
            p2.y = _points[i+1].y;

            tmp = neighbor(p1, p2);

            if(tmp > 1)
            {
                draw_line(w, p1, p2, w1, h1, cimg);
            }
        }
    }
}
```



```

    else if(tmp == 1) -> draw_point(w, p1.x, p1.y, h1, cimg);
}

i = _numPoints-1;
p1.x = _points[i].x;
p1.y = _points[i].y;

i = 0;
p2.x = _points[i].x;
p2.y = _points[i].y;

tmp = neighbor(p1, p2);
if(tmp > 1)
{
    draw_line(w, p1, p2, w1, h1, cimg);
}
else if(tmp == 1) uw -> draw_point(w, p1.x, p1.y, w1, h1, cimg);

delete uw;
}
}

void Points::draw_line(Widget w, Point p1, Point p2, int w1, int h1, ColorImage *cimg)
{
    float dx = fabsf(p2.x - p1.x);
    float dy = fabsf(p2.y - p1.y);

    Point s1, s2, t1, t2;

    if(dy >= dx)
    {
        s1.x = p1.x - 1;
        s1.y = p1.y;
        s2.x = p2.x - 1;
        s2.y = p2.y;
        t1.x = p1.x + 1;
        t1.y = p1.y;
        t2.x = p2.x + 1;
        t2.y = p2.y;
    }
    else
    {
        s1.x = p1.x;
        s1.y = p1.y - 1;
        s2.x = p2.x;
        s2.y = p2.y - 1;
        t1.x = p1.x;
        t1.y = p1.y + 1;
        t2.x = p2.x;
        t2.y = p2.y + 1;
    }

    int i, num;
    Utility_Widget *uw = new Utility_Widget();
    Point *p = new Point[1000];

    num = get_points_in_between(p1, p2, p);
    if(num > 0)
    {
        for(i=0; i<num; i++)
            uw -> draw_point(w, p[i].x, p[i].y, w1, h1, cimg);
    }

    num = get_points_in_between(s1, s2, p);
    if(num > 0)
    {

```

```

        for(i=0; i<num; i++)
            uw -> draw_point(p[i].x, p[i].y, w1, h1, cimg);
    }

    num = get_points_in_between(t1, t2, p);
    if(num > 0)
    {
        for(i=0; i<num; i++)
            uw -> draw_point(w, p[i].x, p[i].y, w1, h1, cimg);
    }

    delete uw;
    delete p;
}

void Points::draw(Widget w, GC gc)
{
    if(_numPoints > 0)
    {
        int i, tmp;
        Point p1, p2;

        Utility_Widget *u = new Utility_Widget();

        for(i=0; i<(_numPoints-1); i++)
        {
            p1.x = _points[i].x;
            p1.y = _points[i].y;
            p2.x = _points[i+1].x;
            p2.y = _points[i+1].y;

            tmp = neighbor(p1, p2);

            if(tmp > 1)
                u -> draw_line(w, gc, p1.x, p1.y,
                               p2.x, p2.y);
            else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);
        }
        i = _numPoints-1;
        p1.x = _points[i].x;
        p1.y = _points[i].y;

        i = 0;
        p2.x = _points[i].x;
        p2.y = _points[i].y;

        tmp = neighbor(p1, p2);
        if(tmp > 1)
            u -> draw_line(w, gc, p1.x, p1.y, p2.x, p2.y);
        else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

        delete u;
    }
}

void Points::draw(Widget w, int color)
{
    if(_numPoints > 0)
    {
        int i, tmp;
        Point p1, p2;

        Utility_Widget *u = new Utility_Widget();

        GC gc = u -> get_GC(w, color);
    }
}

```

```

for(i=0; i<(_numPoints-1); i++)
{
    p1.x = _points[i].x;
    p1.y = _points[i].y;
    p2.x = _points[i+1].x;
    p2.y = _points[i+1].y;

    tmp = neighbor(p1, p2);

    if(tmp > 1)
        u -> draw_line(w, gc, p1.x, p1.y,
            p2.x, p2.y );
    else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);
}
i = _numPoints-1;
p1.x = _points[i].x;
p1.y = _points[i].y;

i = 0;
p2.x = _points[i].x;
p2.y = _points[i].y;

tmp = neighbor(p1, p2);
if(tmp > 1)
    u -> draw_line(w, gc, p1.x, p1.y, p2.x, p2.y);
else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

XtReleaseGC(w, gc);
delete u;
}
}

```

```

void Points::draw_keyPoints(Widget w)
{

```

```

    if(_numPoints > 0)
    {

```

```

        int i, tmp;
        Point p1, p2;

```

```

        Utility_Widget *u = new Utility_Widget();

```

```

        GC gc = u -> get_GC(w, COLOR_YELLOW);

```

```

        int i0 = 0;

```

```

        int k = 0;

```

```

        for(i=i0; i<(_numPoints-1); i++)
        {

```

```

            ++k;

```

```

            p1.x = _points[i].x;

```

```

            p1.y = _points[i].y;

```

```

            p2.x = _points[i+1].x;

```

```

            p2.y = _points[i+1].y;

```

```

            tmp = neighbor(p1, p2);

```

```

            if(tmp > 1)

```

```

                u -> draw_line(w, gc, p1.x, p1.y,

```

```

                    p2.x, p2.y );

```

```

            else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

```

```

            if(k == 5) break;

```

```

        }

```

```

        k = 0;

```

```

        for(i=_numPoints-6; i<(_numPoints-1); i++)

```

```

        if(i >= 0)

```

```

        {

```

```

++k;
p1.x = _points[i];
p1.y = _points[i].y;
p2.x = _points[i+1].x;
p2.y = _points[i+1].y;

tmp = neighbor(p1, p2);

if(tmp > 1)
    u -> draw_line(w, gc, p1.x, p1.y,
        p2.x, p2.y );
else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

if(k == 5) break;
}
XtReleaseGC(w, gc);

gc = u -> get_GC(w, COLOR_GREEN);
i0 = _numPoints/4;
k = 0;
for(i=i0-5; i<(_numPoints-1); i++)
if(i >= 0)
{
    ++k;
    p1.x = _points[i].x;
    p1.y = _points[i].y;
    p2.x = _points[i+1].x;
    p2.y = _points[i+1].y;

    tmp = neighbor(p1, p2);

    if(tmp > 1)
        u -> draw_line(w, gc, p1.x, p1.y,
            p2.x, p2.y );
    else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

    if(k == 10) break;
}
XtReleaseGC(w, gc);

gc = u -> get_GC(w, COLOR_BLUE);
i0 = _numPoints/2;
k = 0;
for(i=i0-5; i<(_numPoints-1); i++)
if(i >= 0)
{
    ++k;
    p1.x = _points[i].x;
    p1.y = _points[i].y;
    p2.x = _points[i+1].x;
    p2.y = _points[i+1].y;

    tmp = neighbor(p1, p2);

    if(tmp > 1)
        u -> draw_line(w, gc, p1.x, p1.y,
            p2.x, p2.y );
    else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

    if(k == 10) break;
}
XtReleaseGC(w, gc);

gc = u -> get_GC(w, COLOR_WHITE);
i0 = _numPoints/4*3;
k = 0;

```

```

for(i=i0-5; i<(_numPoints-1); i++)
if(i >= 0)
{
    ++k;
    p1.x = _points[i].x;
    p1.y = _points[i].y;
    p2.x = _points[i+1].x;
    p2.y = _points[i+1].y;

    tmp = neighbor(p1, p2);

    if(tmp > 1)
        u -> draw_line(w, gc, p1.x, p1.y,
            p2.x, p2.y );
    else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);

    if(k == 10) break;
}
XtReleaseGC(w, gc);

delete u;
}
}

void Points::draw_noloop(Widget w, GC gc)
{
    if(_numPoints > 0)
    {
        int i, tmp;
        Point p1, p2;

        Utility_Widget *u = new Utility_Widget();

        for(i=0; i<(_numPoints-1); i++)
        {
            p1.x = _points[i].x;
            p1.y = _points[i].y;
            p2.x = _points[i+1].x;
            p2.y = _points[i+1].y;

            tmp = neighbor(p1, p2);

            if(tmp > 1)
                u -> draw_line(w, gc, p1.x, p1.y,
                    p2.x, p2.y );
            else if(tmp == 1) u -> draw_point(w, gc, p1.x, p1.y);
        }
        delete u;
    }
}

void Points::show_info()
{
    if(_numPoints > 0)
    {
        printf("Points:: Info \n_numPoints: %d\n", _numPoints);
        for(int i=0; i<_numPoints; i++)
            printf(" <%d:  %f %f>\n", i, _points[i].x, _points[i].y);
    }
}

XPoint *Points::get_XPoint()
{
    if(_numPoints > 0)
    {

```

```

XPoint *xpoint = new XPoint[_numPoints];
for(int i=0; i<_numPoints; i++)
{
    xpoint[i].x = _points[i].x;
    xpoint[i].y = _points[i].y;
}
return xpoint;
}
return NULL;
}

```

```

void Points::translation1(int xc, int yc)
{
    if(_numPoints > 0)
    for(int i=0; i<_numPoints; i++)
    {
        _points[i].x = (float)xc + _points[i].x;
        _points[i].y = (float)yc - _points[i].y;
    }
}

```

```

Points *Points::create()
{
    Points *p = new Points();

    p->_numPoints = _numPoints;
    if(_numPoints > 0)
    for(int i=0; i<_numPoints; i++)
    {
        p->_points[i].x = _points[i].x;
        p->_points[i].y = _points[i].y;
    }
    return p;
}

```

```

void Points::translation2(int dx, int dy)
{
    if(_numPoints > 0)
    for(int i=0; i<_numPoints; i++)
    {
        _points[i].x += (float)dx;
        _points[i].y += (float)dy;
    }
}

```

```

int Points::closest(int x, int y)
{
    if(_numPoints > 0)
    {
        float d2;
        Utility_Math *u = new Utility_Math();

        int k = 0;
        float d = u -> distance(_points[0].x, _points[0].y, float(x), float(y));

        for(int i=1; i<_numPoints; i++)
        {
            d2 = u -> distance(_points[i].x, _points[i].y, float(x), float(y));
            if(d2 < d) {k = i; d = d2;}
        }

        delete u;
        return k;
    }
    else return -1;
}

```

```

}

void Points::zoom(float zoom, int x, int y)
{
    if(_numPoints > 0)
        for(int i=0; i<_numPoints; i++)
        {
            _points[i].x = (float)x + _points[i].x/zoom ;
            _points[i].y = (float)y + _points[i].y/zoom ;
        }
}

Points *Points::get_Points(float zoom, int x, int y)
{
    //printf(" get_Points  %d\n", _numPoints);
    if(_numPoints > 0)
    {
        Points *p = new Points;
        p->_numPoints = _numPoints;
        for(int i=0; i<_numPoints; i++)
        {
            p->_points[i].x = (float)x + _points[i].x/zoom ;
            p->_points[i].y = (float)y + _points[i].y/zoom ;
        }
        return p;
    }
    else return NULL;
}

Points *Points::inverse_get_Points(float zoom, int x, int y)
{
    //printf(" Points::inv_zoom %f %d %d\n", zoom, x, y);

    if(_numPoints > 0)
    {
        Points *p = new Points;
        p->_numPoints = _numPoints;
        for(int i=0; i<_numPoints; i++)
        {
            p->_points[i].x = ((float)_points[i].x - (float)x)*zoom ;
            p->_points[i].y = ((float)_points[i].y - (float)y)*zoom ;
        }
        return p;
    }
    else return NULL;
}

void Points::inverse_zoom(float zoom, int x, int y)
{
    printf(" Points::inv_zoom %f %d %d\n", zoom, x, y);

    if(_numPoints > 0)
        for(int i=0; i<_numPoints; i++)
        {
            _points[i].x = ((float)_points[i].x - (float)x)*zoom ;
            _points[i].y = ((float)_points[i].y - (float)y)*zoom ;
        }
}

int Points::get_minmax(float *min_x, float *max_x, float *min_y, float *max_y)
{
    if(_numPoints > 0)
    {
        *min_x = _points[0].x;
        *min_y = _points[0].y;
        *max_x = _points[0].x;
    }
}

```

```

    *max_y = _points[0].y;
    for(int i=1; i<_numPoints; i++)
    {
        if(_points[i].x < *min_x) *min_x = _points[i].x;
        if(_points[i].y < *min_y) *min_y = _points[i].y;
        if(_points[i].x > *max_x) *max_x = _points[i].x;
        if(_points[i].y > *max_y) *max_y = _points[i].y;
    }
    return 1;
}
else return 0;
}

int Points::neighbor(Point p1, Point p2)
{
    Utility_Math *u = new Utility_Math();

    int dx = u->int_t(p1.x - p2.x);
    int dy = u->int_t(p1.y - p2.y);

    delete u;

    if(dx < 0) dx = -dx;
    if(dy < 0) dy = -dy;

    if(dx == 0 && dy == 0) return 0;
    if(dx <= 1 && dy <= 1) return 1;
    else return 2;
}

int Points::get_points_in_between(Point p1, Point p2, Point *point)
{
    float dx = p2.x - p1.x;
    float dy = p2.y - p1.y;

    float x, y;
    int k = 0;
    float s;

    if( fabsf(dx) > fabsf(dy) )
    {
        s = dy/dx;

        if(p1.x > p2.x)
            for(x = (p1.x - 1); x > p2.x; x--)
            {
                point[k].x = x;
                point[k].y = p1.y - s * (p1.x - x);
                ++k;
            }
        else if(p2.x > p1.x)
            for(x = (p1.x + 1); x < p2.x; x++)
            {
                point[k].x = x;
                point[k].y = p1.y - s * (p1.x - x);
                ++k;
            }
    }
    else
    {
        s = dx/dy;
        //printf(" s %f \n", s);

        if(p1.y > p2.y)

```



```

    for(y = (p1.y - 1); p2.y; y--)
    {
        point[k].y = y;
        point[k].x = p1.x - s * (p1.y - y);
        ++k;
    }
    else if(p2.y > p1.y)
    for(y = (p1.y + 1); y < p2.y; y++)
    {
        point[k].y = y;
        point[k].x = p1.x - s * (p1.y - y);
        ++k;
    }
}
return k;
}

void Points::fill()
{
    int i, j, num, tmp;
    Point p1, p2;
    Point *tmp_point;

    Point *point2;
    int num2;

    tmp_point = new Point[1000];

    num2 = _numPoints;
    point2 = new Point[num2];

    for(i=0; i<num2; i++)
    {
        point2[i].x = _points[i].x;
        point2[i].y = _points[i].y;
    }

    clear();

    for(i=0; i<(num2-1); i++)
    {
        p1.x = point2[i].x;
        p1.y = point2[i].y;
        p2.x = point2[i+1].x;
        p2.y = point2[i+1].y;

        tmp = neighbor(p1, p2);

        if(tmp > 1)
        {
            num = get_points_in_between(p1, p2, tmp_point);
            add(p1.x, p1.y);

            if(num > 0)
            {
                for(j=0; j<num; j++)
                {
                    add(tmp_point[j].x, tmp_point[j].y);
                }
            }
        }
        else if(tmp == 1) add(p1.x, p1.y);
        else if(tmp == 0) i++;
    }
}

```

```

i = num2-1;
p1.x = point2[i].x;
p1.y = point2[i].y;

i = 0;
p2.x = point2[i].x;
p2.y = point2[i].y;

tmp = neighbor(p1, p2);

if(tmp > 1)
{
    num = get_points_in_between(p1, p2, tmp_point);
    add(p1.x, p1.y);

    if(num > 0)
    {
        for(j=0; j<num; j++)
            add(tmp_point[j].x, tmp_point[j].y);
    }
}
else if(tmp == 1) add(p1.x, p1.y);

delete point2;
delete tmp_point;
}

void Points::to_File(FILE *fp)
{
    fprintf(fp, " %d\n", _numPoints);
    if(_numPoints > 0)
    {
        for(int i=0; i<_numPoints; i++)
        {
            fprintf(fp, "   %f %f \n", _points[i].x, _points[i].y);
        }
    }
}

void Points::from_ContourFile(FILE *fp)
{
    float tmp;
    fscanf(fp, "%d %f", &_numPoints, &tmp);
    if(_numPoints > 0)
    {
        for(int i=0; i<_numPoints; i++)
        {
            fscanf(fp, "%f %f %f %f %f\n",
                &_points[i].x, &_points[i].y, &tmp, &tmp, &tmp);
        }
    }
}

void Points::to_ContourFile(FILE *fp)
{
    fprintf(fp, "%d\n1.0\n", _numPoints);
    if(_numPoints > 0)
    {
        for(int i=0; i<_numPoints; i++)
        {
            fprintf(fp, "%f %f 0.541196 0.541196 0.990000\n", _points[i].x, _points[i].y);
        }
    }
}

```

```
fprintf(fp, "\n      here {\n");
fprintf(fp, "      radius 0.5\n");
fprintf(fp, "      } \n");
fprintf(fp, "    }\n");

}

fprintf(fp, " \n }\n");
}

void Points::from_File(FILE *fp)
{
    fscanf(fp, "%d", &_numPoints);
    if(_numPoints > 0)
    {
        for(int i=0; i<_numPoints; i++)
        {
            fscanf(fp, "%f %f", &(_points[i].x), &(_points[i].y));
        }
    }
}
```

```
#include "Utility_Widget.h"
#include "ROIMedDrawingArea.h"
#include "ImgAlloc.h"
```

```
ROI::ROI(Widget w, int color)
```

```
{
    _widget = w;

    _draw_status = FALSE;
    _show_status = FALSE;

    Utility_Widget *u = new Utility_Widget();
    _gc = u -> get_GC(w, color);
    delete u;

    _area = NULL;
    _areaOrg = NULL;
}
```

```
ROI::~ROI()
```

```
{
    if(_gc != NULL) XtReleaseGC(_widget, _gc);

    Utility_Vision *u = new Utility_Vision();
    if(_area != NULL) u -> freeImg(_area);
    if(_areaOrg != NULL) u -> freeImg(_area);
    delete u;
}
```

```
void ROI::draw_img()
```

```
{
    int x, y, w, h;
    get_BoundingBox(&x, &y, &w, &h);
    ((ROIMedDrawingArea *)_roiView) -> copyArea(x-1, y-1, w+4, h+4);
}
```

```
void ROI::AcceptROI()
```

```
{
    ((ROIMedDrawingArea *)_roiView) -> AcceptROI();
}
```

```
void ROI::set_area()
```

```
{
    if(!_draw_status) finished(0, 0);
    fill();

    if(_area != NULL)
    {
        Utility_Vision *uv = new Utility_Vision();
        uv -> freeImg(_area);
        delete uv;
    }

    Utility_Widget *uw = new Utility_Widget();
    _area = uw -> get_mask(_widget, _width, _height);
    delete uw;

    ((ROIMedDrawingArea *)_roiView) -> display();
}
```

```
void ROI::set_areaOrg(float zoom)
```

```
{
    if(_area != NULL)
    {
```

```

int w = int( float(_width) / zoom );
int h = int( float(_height) / zoom );

printf("\n\nset_areaOrg %d %d  %f\n", w, h, zoom);

int i0, i1, j1;

i0 = int(zoom/2.0);

_areaOrg = alloc_img(w, h);
for(int i=0; i<h; i++)
for(int j=0; j<w; j++)
{
    i1 = int( (float)i * zoom ) + i0;
    j1 = int( (float)j * zoom ) + i0;
    _areaOrg[i][j] = _area[i1][j1];
}
}

unsigned char **ROI::copyArea()
{
    unsigned char **area = NULL;

    if(_area != NULL)
    {
        area = alloc_img(_width, _height);
        for(int i=0; i<_height; i++)
        for(int j=0; j<_width; j++)
            area[i][j] = _area[i][j];
    }
    return area;
}

```

```

#include "ROIS.h"
#include <stdio.h>
#include <Vk/VkComponent.h>
#include <string.h>
#include "ROI.h"
#include "Utility.h"
#include "Cylinder.h"

```

```

ROIS::ROIS(int numFrames)
{
    _numFrames = numFrames;
    _ROI = new ROI_Struct[_numFrames];

    int i, j;
    for(i=0; i<_numFrames; i++)
    {
        _ROI[i]._numROIs = 0;
        for(j=0; j<10; j++)
            _ROI[i]._ROI_OBJ[j]._points = NULL;
    }
}

```

```

ROIS::~~ROIS()
{
    int i, j;

    if(_ROI != NULL)
    {
        for(i=0; i<_numFrames; i++)
        {
            for(j=0; j<10; j++)
                if(_ROI[i]._ROI_OBJ[j]._points != NULL)
                    delete _ROI[i]._ROI_OBJ[j]._points;
        }
        delete _ROI;
    }
}

```

```

void ROIS::remove(int img_number, int roi_number)
{
    int n = _ROI[img_number]._numROIs;

    printf("ROIS::remove %d      %d / %d\n", img_number, roi_number, n);

    if(n != 0)
    {
        if(roi_number >= 0 && roi_number < n)
        {
            delete _ROI[img_number]._ROI_OBJ[roi_number]._points;

            if(roi_number != (n-1))
                for(int i=roi_number+1; i<n; i++)
                {
                    sprintf(_ROI[img_number]._ROI_OBJ[i-1]._name, "%s",
                        _ROI[img_number]._ROI_OBJ[i]._name);

                    _ROI[img_number]._ROI_OBJ[i-1]._points =
                        _ROI[img_number]._ROI_OBJ[i]._points;
                }
            --(_ROI[img_number]._numROIs);
        }
    }
}

```

```

void ROIS::add(int img_number, char *name, Points *p)
{

```

```

int n = _ROI[img_number]._numROIs;
int flag = n;
int i;

if(n != 0)
{
    for(i=0; i<n; i++)
        if(strcmp(_ROI[img_number]._ROI_OBJ[i]._name, name) == 0)
        {
            flag = i;
            break;
        }
}

if(flag == n) ++(_ROI[img_number]._numROIs);
else delete _ROI[img_number]._ROI_OBJ[flag]._points;

sprintf(_ROI[img_number]._ROI_OBJ[flag]._name, "%s", name);

_ROI[img_number]._ROI_OBJ[flag]._points = p;
}

void ROIS::to_File()
{
    int i, j, k, n, x, y;
    FILE *fp;

    fp = fopen("ROIS.dat", "w");

    fprintf(fp, "%d\n", _numFrames);
    for(k=0; k<_numFrames; k++)
    {
        n = _ROI[k]._numROIs;
        fprintf(fp, "%d %d\n", k, n);
        if(n != 0)
        {
            for(i=0; i<n; i++)
            {
                fprintf(fp, " %s", _ROI[k]._ROI_OBJ[i]._name);

                if(_ROI[k]._ROI_OBJ[i]._points != NULL)
                {
                    _ROI[k]._ROI_OBJ[i]._points -> to_File(fp);
                }
                else fprintf(fp, "\n");
            }
        }
    }
    fclose(fp);
}

void ROIS::to_File(float thickness, float pX, float pY)
{
    int i, j, k, n, x, y;
    FILE *fp;

    fp = fopen("ROIS.FEM", "w");

    i = 0;
    for(k=0; k<_numFrames; k++)
    {
        n = _ROI[k]._numROIs;
        if(n != 0) ++i;
    }
}

```

```

        if(_ROI[k]._ROI_OBJ[i]._points != NULL)
        {
            cylinder -> add( k*thickness, pX, pY,
                _ROI[k]._ROI_OBJ[i]._points );
            printf(" ROIS:: _numROIs == 2(i=1)  k = %d\n", k);
        }
    }
    if(cylinder->_numFrames > 1) cylinder -> to_ivFile(fp);

    fprintf(fp, " }\n");

    fclose(fp);
    delete u;
    delete cylinder;
    printf(" to-ivFile is done\n");

}
*/

void ROIS::from_File(char *fname)
{
    int    tmp, i, j, k, n, x, y;
    FILE   *fp;

    if( (fp = fopen(fname, "r")) == NULL )
        return;

    fscanf(fp, "%d", &_numFrames);

    if(_numFrames > 0)
    for(k=0; k<_numFrames; k++)
    {
        fscanf(fp, "%d %d", &tmp, &n);
        _ROI[k]._numROIs = n;
        if(n != 0)
        {
            for(i=0; i<n; i++)
            {
                fscanf(fp, "%s", _ROI[k]._ROI_OBJ[i]._name);
                _ROI[k]._ROI_OBJ[i]._points = new Points();
                _ROI[k]._ROI_OBJ[i]._points -> from_File(fp);
            }
        }
    }

    fclose(fp);
}

```


User: meide
Host: phoenix
Class: phoenix
Job: Polygon.C

150

```
#include "TwoLines.h"
```

```
#include "Utility_Math.h"
```

```
#include "Utility_Widget.h"
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
TwoLines::TwoLines(Widget w, int mx, int my)
```

```
{
    _widget = w;
    Utility_Widget *u = new Utility_Widget();
    _gc = u -> get_xorGC(_widget);

    _x1 = 0;
    _x2 = mx;

    _max_x = mx;
    _max_y = my;
    delete u;
}
```

```
TwoLines::~~TwoLines()
```

```
{
    if(_gc != NULL) XtReleaseGC(_widget, _gc);
}
```

```
void TwoLines::set(float center, float width)
```

```
{
    _x1 = center - width/2.0;
    _x2 = center + width/2.0;

    if(_x1 < 0) _x1 = 0;
    else if(_x1 > (float)_max_x) _x1 = (float)_max_x;

    if(_x2 < 0) _x2 = 0;
    else if(_x2 > (float)_max_x) _x2 = (float)_max_x;
}
```

```
void TwoLines::draw()
```

```
{
    Utility_Math *um = new Utility_Math();
    int x1 = um->int_t(_x1);
    int x2 = um->int_t(_x2);
    delete um;

    Utility_Widget *uw = new Utility_Widget();
    uw -> draw_line(_widget, _gc, x1, 0, x1, _max_y);
    uw -> draw_line(_widget, _gc, x2, 0, x2, _max_y);
    delete uw;
}
```

```
void TwoLines::init(int x)
```

```
{
    if(_x1 == _x2)
    {
        if(x == _x2) _status = TWOLINES_MOVE;
        else if(x > _x2) _status = TWOLINES_RIGHT;
        else _status = TWOLINES_LEFT;
    }
    else if(x == _x1) _status = TWOLINES_INCREASE_RIGHT;
    else if(x == _x2) _status = TWOLINES_INCREASE_LEFT;
    else if(x > _x1 && x < _x2) _status = TWOLINES_MOVE;
    else if(x < _x1) _status = TWOLINES_LEFT;
    else if(x > _x2) _status = TWOLINES_RIGHT;
}
```

```

    _current_x = (float)x;
}

void TwoLines::draw(int x)
{
    float dx = (float)x - _current_x;
    switch (_status)
    {
        case TWOLINES_INCREASE_LEFT:
            _x1 -= dx;
            _x2 += dx;
            break;
        case TWOLINES_INCREASE_RIGHT:
            _x1 += dx;
            _x2 -= dx;
            break;
        case TWOLINES_MOVE:
            _x1 += dx;
            _x2 += dx;
            break;
        case TWOLINES_LEFT:
            _x1 += dx;
            break;
        case TWOLINES_RIGHT:
            _x2 += dx;
            break;
        default:
            break;
    }

    if(_x1 < 0) _x1 = 0; // _x1 >= 0
    if(_x2 < 0) _x2 = 0; // _x2 >= 0
    if(_x2 > (float)_max_x) _x2 = (float)_max_x; // _x2 <= _max_x
    if(_x1 > _x2) _x1 = _x2; // _x1 <= _x2

    draw();

    _current_x = (float)x;
}

```

```
#ifndef ANIMATE_C
#define ANIMATE_C
```

```
#include "Animate.h"
```

```
#include "LineDrawingArea.h"
#include "Utility_Widget.h"
#include <stdio.h>
```

```
void animation()
```

```
{
    //printf(" in Animate \n");

    if(_animate->_pixmap != NULL && _animate->_pixmap[_animate->_img_number] != NULL)
        XCopyArea(XtDisplay(_animate->_widget), _animate->_pixmap[_animate->_img_number],
            XtWindow(_animate->_widget), _animate->_gc, 0, 0, _animate->_width, _animate->_height);

    //printf(" in Animate 1 \n");

    if(_animate->_lWave != NULL && _animate->_wave_number != 0)
        ((LineDrawingArea *) (_animate->_lWave)) -> draw_onePoint(_animate->_wave_number,
            _animate->_lWaveGC);

    //printf(" in Animate 1A \n");

    if(_animate->_rWave != NULL && _animate->_wave_number != 0)
        ((LineDrawingArea *) (_animate->_rWave)) -> draw_onePoint(_animate->_wave_number,
            _animate->_rWaveGC);

    if(_animate->_ivview != NULL && _animate->_soiv[_animate->_img_number]._iv != NULL)
        _animate->_ivview -> setSceneGraph((_animate->_soiv[_animate->_img_number]._iv));

    //printf(" in Animate 1B \n");

    ++(_animate->_img_number);
    if(_animate->_img_number == _animate->_num_imgs)
        _animate->_img_number = 0;

    //printf(" in Animate 1C \n");

    if(_animate->_rWave != NULL || _animate->_lWave != NULL)
    {
        //printf(" in Animate 1D\n");

        ++(_animate->_wave_number);
        //printf(" in Animate 1E\n");

        if(_animate->_wave_number % _animate->_num_imgs != _animate->_img_number)
        {
            int tmp = _animate->_wave_number / _animate->_num_imgs;
            _animate->_wave_number = _animate->_num_imgs * tmp + _animate->_img_number;
            //printf(" in Animate 1F\n");
        }

        //printf(" in Animate 2    %d  %d\n", _animate->_wave_number, _animate->_num_waves);

        if(_animate->_wave_number == _animate->_num_waves)
        {
            //printf(" in Animate 3 \n");

            _animate->_wave_number = 0;

            Utility_Widget *uw = new Utility_Widget();

            if(_animate->_lWave != NULL)
```

```

{
    if(_animate->_lWaveColor == COLOR_RED) _animate->_lWaveColor = COLOR_GREEN;
    else _animate->_lWaveColor = COLOR_RED;

    XtReleaseGC(_animate->_widget, _animate->_lWaveGC);

    _animate->_lWaveGC = uw->get_GC(_animate->_widget, _animate->_lWaveColor);
}

if(_animate->_rWave != NULL)
{
    if(_animate->_rWaveColor == COLOR_RED) _animate->_rWaveColor = COLOR_GREEN;
    else _animate->_rWaveColor = COLOR_RED;

    XtReleaseGC(_animate->_widget, _animate->_rWaveGC);

    _animate->_rWaveGC = uw->get_GC(_animate->_widget, _animate->_rWaveColor);
}

delete uw;
}
//printf(" in Animate 4\n");
}

//printf(" in Animate 5    %d\n", _animate->_time_out);
if(_animate->_time_out == 1)
{
    if(_animate->_firsttime != 1) XtRemoveTimeOut(_animate->_id);
    _animate->_id = XtAppAddTimeOut(XtWidgetToApplicationContext(_animate->_widget),
    _animate->_msec, (XtTimerCallbackProc)animation, NULL);
    //printf(" in Animate 5    %d %d\n", _animate->_time_out, _animate->_msec);
    _animate->_firsttime = 0;
}
}

#endif

```

```
#ifndef PROGRESS_C
#define PROGRESS_C
```

154

```
#include "Progress.h"
```

```
#include "Animate.h"
```

```
#include "Utility.h"
```

```
#include "Utility_Vision.h"
```

```
#include "ObjectManager.h"
```

```
#include <stdio.h>
```

```
void Progress_Animate2D()
```

```
{
    ImgGE *imgGE = progress->_objMag -> get_ImgGE(progress->curr, progress->_objMag->
        progress->_objMag->msgsRight.img_pcmra_type, NULL);

    short **img;
    Utility_Vision *uv = new Utility_Vision();

    if(progress->_objMag->msgsRight.img_select == RIGHT_IMG_ROI && progress->_objMag->
        && progress->_objMag->_imgView -> _ROI -> _draw_status)
    {
        //imgGE = progress->_objMag -> get_ImgGE2(progress->curr, imgGE);

int x1 = int( float(progress->_objMag->msgsLeft.roi_x)/progress->_objMag->msgsLeft.roi_w);
        int y1 = int( float(progress->_objMag->msgsLeft.roi_y)/progress->_objMag->msgsLeft.roi_h);
        int w1 = int( float(progress->_objMag->msgsLeft.roi_w)/progress->_objMag->msgsLeft.roi_h);
        int h1 = int( float(progress->_objMag->msgsLeft.roi_h)/progress->_objMag->msgsLeft.roi_w);

        if(progress->_objMag->msgsRight.img_type == IMAGE_PCMRA &&
            progress->_objMag->msgsRight.img_pcmra_type == PCMRA_VELOCITY &&
            progress->_objMag->msgsRight.velocity_select == VELOCITY_ROIMASKED &&
            progress->_objMag->_imgView2 -> _ROI != NULL)
            img = uv -> get_ROI(imgGE-> get_imgdata(), x1, y1, w1, h1,
                progress->_objMag->_imgView2 -> _ROI -> _areaOrg);
        else
img = imgGE-> get_imgdata(), x1, y1, w1, h1, progress->_objMag->_imgView2 -> _ROI -> _areaOrg;

        //img = imgGE -> get_imgdata();
    }
    else img = imgGE-> get_imgdata();

    _animate->_pixmap[progress->curr - progress->_objMag->msgsLoaded.img_start] =
        progress->_objMag->_imgView2 -> get_pixmap(img);

    ++(progress->curr);
    printf(" Progress::  %d \n", progress->curr);
    progress -> window -> update_percent(progress->curr - progress->_objMag->msgsLoaded.img_start,
        progress->_objMag->msgsLoaded.img_end - progress->_objMag->msgsLoaded.img_start)

    uv -> freeShimg(img);
    img = NULL;

    delete imgGE;
    delete uv;

    if(progress -> window -> get_status()) progress->time_out = 0;

    if(progress->time_out == 1 && progress->curr <= progress->_objMag->msgsLoaded.img_start)
    {
        printf(" Animate\n");
        if(progress->firsttime != 1) XtRemoveTimeOut(progress->id);
        progress->id = XtAppAddTimeOut(XtWidgetToApplicationContext(progress->widget),
            progress->msec, (XtTimerCallbackProc)Progress_Animate2D, NULL);
        progress->firsttime = 0;
    }
}
```



```

else
{
    delete imgGE;
    delete uv;
    return;
}

```

```
Utility_3D *u3D = new Utility_3D();
```

```

u3D -> to_ivFileAnimateRot(progress->curr - progress->_objMag->msgsLoaded.img_start,
    w, h, img, &(progress->_objMag->msgsRight.ratio3D),
    progress->_objMag->msgsRight.camera, &(progress->_objMag->msgsRight.YPos3D),
    progress->_objMag->msgsRight.flow3DDir);

```

```
delete u3D;
```

```
SoInput *sceneFile = new SoInput();
```

```
sceneFile->openFile("tmp.iv");
```

```

_animate->_soiv[progress->curr - progress->_objMag->msgsLoaded.img_start]._iv = 1;
_animate->_soiv[progress->curr - progress->_objMag->msgsLoaded.img_start]._iv ->

```

```
++(progress->curr);
```

```

progress -> window -> update_percent(progress->curr - progress->_objMag->msgsLoaded.img_start,
    progress->_objMag->msgsLoaded.img_end - progress->_objMag->msgsLoaded.img_start)

```

```

uv -> freeShimg(img);
img = NULL;

```

```

delete sceneFile;
delete imgGE;
delete uv;

```

```
if(progress -> window -> get_status()) progress->time_out = 0;
```

```

if(progress->time_out == 1 && progress->curr <= progress->_objMag->msgsLoaded.img_end)
{
    printf(" Animate\n");
    if(progress->firsttime != 1) XtRemoveTimeOut(progress->id);
    progress->id = XtAppAddTimeOut(XtWidgetToApplicationContext(progress->widget),
        progress->msec, (XtTimerCallbackProc)Progress_Animate3D, NULL);
    progress->firsttime = 0;
}

```

```
else if(_animate->_toBeFinished)
```

```

{
    printf("\n\n Finished Progress:: %d \n", progress->curr);
    XtRemoveTimeOut(progress->id);
    delete progress -> window;
    progress -> window = NULL;

```

```
    progress->_objMag -> start_animate();
```

```
}
```

```
}
```

```
void Progress_AnimateSymphony()
```

```
{
```

```

    ImgGE *imgGE = progress->_objMag -> get_ImgGE(progress->curr, progress->_objMag->
        progress->_objMag->msgsRight.img_pcmra_type, NULL);

```

```

    int    w, h;
    short **img;
    Utility_Vision *uv = new Utility_Vision();

```

```
if(progress->_objMag->msgsRight.img_select == RIGHT_IMG_ROI && progress->_objMag->
```



```

    && progress->_objMag->_imgView -> _ROI -> _draw_status)
{
    int x1 = int( float(progress->_objMag->msgsLeft.roi_x)/progress->_objMag->msgsI
    int y1 = int( float(progress->_objMag->msgsLeft.roi_y)/progress->_objMag->msgsI
    w = int( float(progress->_objMag->msgsLeft.roi_w)/progress->_objMag->msgsLeft.i
    h = int( float(progress->_objMag->msgsLeft.roi_h)/progress->_objMag->msgsLeft.i

    if(progress->_objMag->msgsRight.img_type == IMAGE_PCMRA &&
        progress->_objMag->msgsRight.img_pcmra_type == PCMRA_VELOCITY &&
        progress->_objMag->msgsRight.velocity_select == VELOCITY_FLOWMASKED)
    {
        printf(" Hi *****\n");
        if(progress->_objMag->_imgView2 -> _ROI != NULL &&
            progress->_objMag->_imgView2 -> _ROI -> _areaOrg != NULL)
            imgGE -> set(x1, y1, w, h, imgGE -> get_imgdata(),
                progress->_objMag->_imgView2 -> _ROI -> _areaOrg);
        else
            imgGE -> set(x1, y1, w, h, imgGE -> get_imgdata());

        progress->_objMag->update_Rimg2D(imgGE);

        if(progress->_objMag->msgsRight.roi_mask != NULL)
            imgGE -> set(progress->_objMag->msgsRight.roi_mask,
                progress->_objMag->msgsRight.velocity_ratio);

        img = imgGE-> get_imgdata();
        printf("\n 3D Animation %d\n", progress->curr);
    }

    /*
    if(progress->_objMag->msgsRight.img_type == IMAGE_PCMRA &&
        progress->_objMag->msgsRight.img_pcmra_type == PCMRA_VELOCITY &&
        progress->_objMag->msgsRight.velocity_select == VELOCITY_ROIMASKED &&
        progress->_objMag->_imgView2 -> _ROI != NULL)
        img = uv -> get_ROI(imgGE-> get_imgdata(),x1, y1, w, h,
            progress->_objMag->_imgView2 -> _ROI -> _areaOrg);

    */

    else
        img = uv -> get_ROI(imgGE-> get_imgdata(),x1, y1, w, h);
    //img = uv -> get_ROI(imgGE-> get_imgdata(),x1, y1, w, h);
}
else
{
    w = imgGE-> get_width();
    h = imgGE-> get_height();
    img = imgGE-> get_imgdata();
}

_animate->_pixmap[progress->curr - progress->_objMag->msgsLoaded.img_start] =
    progress->_objMag->_imgView2 -> get_pixmap(img);

Utility_3D *u3D = new Utility_3D();
u3D -> to_ivFileAnimateRot(progress->curr - progress->_objMag->msgsLoaded.img_sta
    w, h, img, &(progress->_objMag->msgsRight.ratio3D),
    progress->_objMag->msgsRight.camera, &(progress->_objMag->msgsRight.YPos3D),
    progress->_objMag->msgsRight.flow3DDir);
delete u3D;

SoInput *sceneFile = new SoInput();
sceneFile->openFile("tmp.iv");
_animate->_soiv[progress->curr - progress->_objMag->msgsLoaded.img_start]._iv = &
_animate->_soiv[progress->curr - progress->_objMag->msgsLoaded.img_start]._iv ->

++(progress->curr);

progress -> window -> update_percent(progress->curr - progress->_objMag->msgsLoade

```

```
progress->_objMag->msgsLoaded.img_end - progress->_objMag->msgsLoaded.img_start)
158
```

```
uv -> freeShimg(img);
img = NULL;
```

```
delete sceneFile;
delete imgGE;
delete uv;
```

```
if(progress -> window -> get_status()) progress->time_out = 0;
```

```
if(progress->time_out == 1 && progress->curr <= progress->_objMag->msgsLoaded.img
{
    if(progress->firsttime != 1) XtRemoveTimeOut(progress->id);
    progress->id = XtAppAddTimeOut(XtWidgetToApplicationContext(progress->widget),
    progress->msec, (XtTimerCallbackProc)Progress_AnimateSymphony, NULL);
    progress->firsttime = 0;
}
```

```
else if(_animate->_toBeFinished)
```

```
{
    XtRemoveTimeOut(progress->id);
    delete progress -> window;
    progress -> window = NULL;
```

```
    progress->_objMag -> start_animate();
```

```
}
```

```
}
```

```
#endif
```

```

////////////////////////////////////
//
// Source file for SwUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "SwUI.h" // Generated header file for this class

#include <Xm/ScrolledW.h>
#include <Xm/Text.h>
#include <Vk/VkResource.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String SwUI::_defaultSwUIResources[] = {

    //---- Start editable code block: SwUI Default Resources

    //---- End editable code block: SwUI Default Resources

    (char*)NULL
};

SwUI::SwUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: Sw constructor 2

    //---- End editable code block: Sw constructor 2

} // End Constructor

```

[illegible]

```
//----- Start editable code block: SwUI create

//----- End editable code block: SwUI create
}

const char * SwUI::className()
{
    return ("SwUI");
}    // End className()

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for Sw
//
// This file is generated by RapidApp 1.2
//
// This class is derived from SwUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "/*----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "Sw.h"
#include <Vk/VkEZ.h>
#include <Xm/ScrolledW.h>
#include <Xm/Text.h>
#include <Vk/VkResource.h>
#include <Vk/VkSimpleWindow.h>

```

```

////////////////////////////////////
// The following non-container elements are created by SwUI and are
// available as protected data members inherited by this class
//
// XmText          _scrolledText
//
////////////////////////////////////

```

```

/*----- Start editable code block: headers and declarations

```

```

/*----- End editable code block: headers and declarations

```

```

/*----- Sw Constructor

```

```

Sw::Sw(const char *name, Widget parent) :
    SwUI(name, parent)
{
    // This constructor calls SwUI(parent, name)
    // which calls SwUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    /*----- Start editable code block: Sw constructor

    /*----- End editable code block: Sw constructor

}    // End Constructor

```



```
// WARNING: This structure is different than that used with 1.1 RapidApp.  
// See the RapidApp release notes for details
```

164

```
struct InterfaceMap {  
    char *resourceName;  
    char *methodName;  
    char *argType;  
    char *definingClass; // Optional, if not this class  
    void (VkCallbackObject::*method)(...); // Reserved, do not set  
};  
  
void *Sw::RegisterSwInterface()  
{  
    // This structure registers information about this class  
    // that allows RapidApp to create and manipulate an instance.  
    // Each entry provides a resource name that will appear in the  
    // resource manager palette when an instance of this class is  
    // selected, the name of the member function as a string,  
    // the type of the single argument to this function, and an  
    // optional argument indicating the class that defines this function.  
    // All member functions must have the form  
    //  
    //     void memberFunction ( Type );  
    //  
    // where "Type" is one of:  
    //     const char *      (Use XmRString)  
    //     Boolean           (Use XmRBoolean)  
    //     int               (Use XmRInt)  
    //     float             (Use XmRFloat)  
    //     No argument       (Use VkRNoArg or "NoArg")  
    //     A filename        (Use VkRFilename or "Filename")  
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")  
    //     A callback        (Use XmRCallback)  
  
    static InterfaceMap map[] = {  
        //----- Start editable code block: SwUI resource table  
  
        // { "resourceName", "setAttribute", XmRString},  
        //----- End editable code block: SwUI resource table  
        { NULL }, // MUST be NULL terminated  
    };  
  
    return map;  
} // End RegisterSwInterface()
```

```
//----- End of generated code
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```



```

////////////////////////////////////
//
// Source file for InfoMainWindow
//
// This class is a subclass of VkSimpleWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Try to restrict any changes to the bodies of functions
// corresponding to menu items, the constructor and destructor.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
// Avoid gratuitous reformatting and other changes that might
// make it difficult to integrate changes made using RapidApp
////////////////////////////////////
#include "InfoMainWindow.h"

#include <Vk/VkApp.h>
#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include "Sw.h"

extern void VkUnimplemented ( Widget, const char * );

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String InfoMainWindow::_defaultInfoMainWindowResources[] = {
    "*title: Image Information",

    //--- Start editable code block: InfoMainWindow Default Resources

    //--- End editable code block: InfoMainWindow Default Resources

    (char*)NULL
};

//--- Class declaration

InfoMainWindow::InfoMainWindow ( const char *name,
                                ArgList args,
                                Cardinal argCount ) :
    VkSimpleWindow ( name, args, argCount )

```

```

{
    // Load any class-defined resources for this object
    setDefaultResources ( baseWidget(), _defaultInfoMainWindowResources );

    // Create the view component contained by this window
    _sw = new Sw ( "sw",mainWindowWidget() );

    XtVaSetValues ( _sw->baseWidget(),
                    XmNwidth, 560,
                    XmNheight, 394,
                    (XtPointer) NULL );

    // Add the component as the main view
    addView ( _sw );

    //----- Start editable code block: InfoMainWindow constructor

    //----- End editable code block: InfoMainWindow constructor

}    // End Constructor

InfoMainWindow::~InfoMainWindow()
{
    delete _sw;
    //----- Start editable code block: InfoMainWindow destructor

    //----- End editable code block: InfoMainWindow destructor
}    // End destructor

const char *InfoMainWindow::className()
{
    return ("InfoMainWindow");
}    // End className()

Boolean InfoMainWindow::okToQuit()
{
    //----- Start editable code block: InfoMainWindow okToQuit

    // This member function is called when the user quits by calling
    // theApplication->terminate() or uses the window manager close protocol
    // This function can abort the operation by returning FALSE, or do some.
    // cleanup before returning TRUE. The actual decision is normally passed on
    // to the view object

    // Query the view object, and give it a chance to cleanup
    return ( _sw->okToQuit() );

    //----- End editable code block: InfoMainWindow okToQuit
}    // End okToQuit()

```

```
////////////////////////////////////  
// The following functions are called from callbacks  
////////////////////////////////////
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for ProgressBB
//
// This file is generated by RapidApp 1.2
//
// This class is derived from ProgressBBUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "ProgressBB.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Frame.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>
#include <Vk/VkSimpleWindow.h>

```

```

extern void VkUnimplemented ( Widget, const char * );

```

```

////////////////////////////////////
// The following non-container elements are created by ProgressBBUI and are
// available as protected data members inherited by this class
//
// XmPushButton          _buttonCancel
// XmLabel                _labelPercent
// XmFrame                _frame
// XmLabel                _labelTitle
//
////////////////////////////////////

```

```

///---- Start editable code block: headers and declarations

```

```

#include "Utility.h"
#include "Utility_Widget.h"
#include "ImgAlloc.h"

```

```

///---- End editable code block: headers and declarations

```

```

///---- ProgressBB Constructor

```

```

ProgressBB::ProgressBB(const char *name, Widget parent) :
    ProgressBBUI(name, parent)
{
    // This constructor calls ProgressBBUI(parent, name)
    // which calls ProgressBBUI::create() to create
    // the widgets for this component. Any code added here

```

```

// is called after the component's interface has been built

//----- Start editable code block: ProgressBB constructor

_cancel = FALSE;
_map = NULL;
_mapImg = NULL;

//----- End editable code block: ProgressBB constructor

} // End Constructor

ProgressBB::ProgressBB(const char *name) :
    ProgressBBUI(name)
{
    // This constructor calls ProgressBBUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used.

    //----- Start editable code block: ProgressBB constructor 2

    _cancel = FALSE;
    _map = NULL;
    _mapImg = NULL;

    //----- End editable code block: ProgressBB constructor 2

} // End Constructor

ProgressBB::~~ProgressBB()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //----- Start editable code block: ProgressBB destructor

    printf("\n\ndelte ProgressBB\n\n");
    if(_mapImg != NULL) free_shimg(_mapImg);
    if(_map != NULL) delete _map;

    //----- End editable code block: ProgressBB destructor

} // End Destructor

const char * ProgressBB::className() // classname
{
    return ("ProgressBB");
} // End className()

void ProgressBB::doButtonCancel ( Widget w, XtPointer callData )
{
    //----- Start editable code block: ProgressBB doButtonCancel

```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData; 170

//--- Comment out the following line when ProgressBB::doButtonCancel is implemented

//::VkUnimplemented ( w, "ProgressBB::doButtonCancel" );

_cancel = TRUE;

//---- End editable code block: ProgressBB doButtonCancel

} // End ProgressBB::doButtonCancel()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *ProgressBB::CreateProgressBB( const char *name, Widget parent )
{
    VkComponent *obj = new ProgressBB ( name, parent );
    return ( obj );
} // End CreateProgressBB

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *ProgressBB::RegisterProgressBBInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char * (Use XmRString)
    // Boolean (Use XmRBoolean)
    // int (Use XmRInt)

```

```
// float           (Use XmRFloat)
// No argument      (Use VkRNoArg or "NoArg")
// A filename       (Use VkRFilename or "Filename")
// An enumeration   (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
// A callback       (Use XmRCallback)
```

```
static InterfaceMap map[] = {
//---- Start editable code block: ProgressBBUI resource table
```

```
    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: ProgressBBUI resource table
    { NULL }, // MUST be NULL terminated
};
```

```
    return map;
} // End RegisterProgressBBInterface()
```

```
//---- End of generated code
```

```
//---- Start editable code block: End of generated code
```

```
void ProgressBB::init(char *msg)
```

```
{
    int    i, j;

    _width = 350;
    _height = 50;

    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelTitle, msg);
    delete uw;

    _percent = 0;

    _mapImg = alloc_shimg(_width, _height);

    for(i=0; i<_height; i++)
    for(j=0; j<_width; j++)
        _mapImg[i][j] = -1000;

    _map = new MedDrawingArea("Progress", _frame, 0);
    _map -> set(_width, _height, _mapImg, VISUAL_COLOR, SCALE_SPLINE, 1.0, 0, 300, 0);

    _map -> show();
    ((DrawingArea *)_map) -> display(0, 0);

    set_percent(_percent);
}
```

```
void ProgressBB::set_title(char *msg)
```

```
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelTitle, msg);
    delete uw;
}
```

```
void ProgressBB::set_percent(int percent)
```

```
{
    char    msg[100];
```

```

    sprintf(msg, "%d %%", _percent);
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelPercent, msg);
    delete uw;
}

void ProgressBB::update_percent(int curr, int num)
{
    if(curr >= 0 && num > 0 && curr <= num)
    {
        float tmp = float(curr)/float(num);

        for(int i=0; i<_height; i++)
            for(int j=0; j<int(_width*tmp) ; j++)
                _mapImg[i][j] = 45;

        _map -> set(_width, _height, _mapImg, VISUAL_COLOR, SCALE_SPLINE, 1.0, 0, 300, 0);
        _map -> display();

        _percent = int(tmp*100.0 + 0.5);
        set_percent(_percent);
    }
}

//---- End editable code block: End of generated code

```



```

////////////////////////////////////
//
// Source file for ProgressBBUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "ProgressBBUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Frame.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String ProgressBBUI::_defaultProgressBBUIResources[] = {
    "*buttonCancel.labelString: Cancel",
    "*labelPercent.labelString: 50 %",
    "*labelTitle.labelString: Label Title",

    //---- Start editable code block: ProgressBBUI Default Resources

    //---- End editable code block: ProgressBBUI Default Resources

    (char*)NULL
};

ProgressBBUI::ProgressBBUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: ProgressBB constructor 2

```

} // End Constructor

```
ProgressBBUI::ProgressBBUI ( const char *name, Widget parent ) : VkComponent ( name )
{
```

```
    //---- Start editable code block: ProgressBB pre-create
```

```
    //---- End editable code block: ProgressBB pre-create
```

```
    // Call creation function to build the widget tree.
```

```
    create ( parent );
```

```
    //---- Start editable code block: ProgressBB constructor
```

```
    //---- End editable code block: ProgressBB constructor
```

} // End Constructor

```
ProgressBBUI::~ProgressBBUI()
```

{

```
    // Base class destroys widgets
```

```
    //---- Start editable code block: ProgressBBUI destructor
```

```
    //---- End editable code block: ProgressBBUI destructor
```

} // End destructor

```
void ProgressBBUI::create ( Widget parent )
```

{

```
    Arg      args[6];
```

```
    Cardinal count;
```

```
    count = 0;
```

```
    // Load any class-defaulted resources for this object
```

```
    setDefaultResources ( parent, _defaultProgressBBUIResources );
```

```
    // Create an unmanaged widget as the top of the widget hierarchy
```

```
    _baseWidget = _progressBB = XtVaCreateWidget ( _name,
                                                    xmBulletinBoardWidgetClass,
                                                    parent,
                                                    XmNresizePolicy, XmRESIZE_GROW,
                                                    (XtPointer) NULL );
```

```
    // install a callback to guard against unexpected widget destruction
```

```
    installDestroyHandler();
```

```
// Create widgets used in this component
// All variables are members of this class
```

175

```
_buttonCancel = XtVaCreateManagedWidget ( "buttonCancel",
                                           xmPushButtonWidgetClass,
                                           _baseWidget,
                                           XmNlabelType, XmSTRING,
                                           XmNx, 274,
                                           XmNy, 158,
                                           XmNwidth, 100,
                                           XmNheight, 40,
                                           (XtPointer) NULL );
```

```
XtAddCallback ( _buttonCancel,
                XmNactivateCallback,
                &ProgressBBUI::doButtonCancelCallback,
                (XtPointer) this );
```

```
_labelPercent = XtVaCreateManagedWidget ( "labelPercent",
                                           xmLabelWidgetClass,
                                           _baseWidget,
                                           XmNlabelType, XmSTRING,
                                           XmNx, 170,
                                           XmNy, 140,
                                           XmNwidth, 37,
                                           XmNheight, 20,
                                           (XtPointer) NULL );
```

```
_frame = XtVaCreateManagedWidget ( "frame",
                                     xmFrameWidgetClass,
                                     _baseWidget,
                                     XmNshadowType, XmSHADOW_ETCHED_IN,
                                     XmNx, 24,
                                     XmNy, 70,
                                     XmNwidth, 350,
                                     XmNheight, 50,
                                     (XtPointer) NULL );
```

```
_labelTitle = XtVaCreateManagedWidget ( "labelTitle",
                                           xmLabelWidgetClass,
                                           _baseWidget,
                                           XmNlabelType, XmSTRING,
                                           XmNx, 70,
                                           XmNy, 30,
                                           XmNwidth, 74,
                                           XmNheight, 20,
                                           (XtPointer) NULL );
```

```
//----- Start editable code block: ProgressBBUI create
```

```
//----- End editable code block: ProgressBBUI create
```

```
}
```

```
const char * ProgressBBUI::className()
{
    return ("ProgressBBUI");
} // End className()
```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void ProgressBBUI::doButtonCancelCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    ProgressBBUI* obj = ( ProgressBBUI * ) clientData;
    obj->doButtonCancel ( w, callData );
}

```

```

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

```

```

void ProgressBBUI::doButtonCancel ( Widget, XtPointer )
{
    // This virtual function is called from doButtonCancelCallback.
    // This function is normally overridden by a derived class.
}

```

```

//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

User: meide
Host: phoenix
Class: phoenix
Job: TwoLines.C

177

```

////////////////////////////////////
//
// Source file for ProgressMainWindow
//
// This class is a subclass of VkSimpleWindow
//
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Try to restrict any changes to the bodies of functions
// corresponding to menu items, the constructor and destructor.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
// Avoid gratuitous reformatting and other changes that might
// make it difficult to integrate changes made using RapidApp
////////////////////////////////////
#include "ProgressMainWindow.h"

#include <Vk/VkApp.h>
#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include "ProgressBB.h"

extern void VkUnimplemented ( Widget, const char * );

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String ProgressMainWindow::_defaultProgressMainWindowResources[] = {
    "*title: Progress Status",

    //---- Start editable code block: ProgressMainWindow Default Resources

    //---- End editable code block: ProgressMainWindow Default Resources

    (char*)NULL
};

//---- Class declaration

ProgressMainWindow::ProgressMainWindow ( const char *name,
                                         ArgList args,
                                         Cardinal argCount ) :
    VkSimpleWindow ( name, args, argCount )

```

```

{
    // Load any class-default resources for this object
    setDefaultResources ( baseWidget(), _defaultProgressMainWindowResources );

    // Create the view component contained by this window
    _progressBB = new ProgressBB ( "progressBB",mainWindowWidget() );

    XtVaSetValues ( _progressBB->baseWidget(),
                    XmNwidth, 397,
                    XmNheight, 208,
                    (XtPointer) NULL );

    // Add the component as the main view
    addView ( _progressBB );

    //----- Start editable code block: ProgressMainWindow constructor
    //----- End editable code block: ProgressMainWindow constructor
}    // End Constructor

ProgressMainWindow::~ProgressMainWindow()
{
    delete _progressBB;
    //----- Start editable code block: ProgressMainWindow destructor

    //----- End editable code block: ProgressMainWindow destructor
}    // End destructor

const char *ProgressMainWindow::className()
{
    return ("ProgressMainWindow");
}    // End className()

Boolean ProgressMainWindow::okToQuit()
{
    //----- Start editable code block: ProgressMainWindow okToQuit

    // This member function is called when the user quits by calling
    // theApplication->terminate() or uses the window manager close protocol
    // This function can abort the operation by returning FALSE, or do some.
    // cleanup before returning TRUE. The actual decision is normally passed on
    // to the view object

    // Query the view object, and give it a chance to cleanup
    return ( _progressBB->okToQuit() );

    //----- End editable code block: ProgressMainWindow okToQuit
}    // End okToQuit()

```

```
////////////////////////////////////  
// The following functions are called from callbacks  
////////////////////////////////////
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```



```

#include "Cylinder.h"
#include "Utility.h"
#include <stdio.h>
#include <math.h>

```

```

Cylinder::Cylinder()
{
    _numFrames = 0;
}

```

```

Cylinder::~~Cylinder()
{
}

```

```

void Cylinder::clear()
{
    _numFrames = 0;
}

```

```

void Cylinder::add(int index_z, Points *p)
{
    _z[_numFrames] = index_z;

    _plane[_numFrames].clear();

    for(int i=0; i<p->_numPoints; i++)
    {
        _plane[_numFrames].add(p->_points[i].x, p->_points[i].y);
    }

    printf("      %d:   %d   %d (%d)\n", _numFrames, index_z, p->_numPoints, _plane[_nu
        ++_numFrames;
}

```

```

void Cylinder::uniform_Points()
{
    if(_numFrames == 0) return;

    float degree = 2;

    int n = int(360.0/degree);
    _numPoints = n;

    int    num;
    Point  *point = new Point[1000];
    float  sita, x, y, xc, yc;

    float pi = 3.141592654;

    int    i, j;
    Utility *u = new Utility();

    for(i=0; i<_numFrames; i++)
    {
        num = _plane[i]._numPoints;
        for(j=0; j<num; j++)
        {
            point[j].x = _plane[i]._points[j].x;
            point[j].y = _plane[i]._points[j].y;
        }
        u->get_point(num, point, &xc, &yc);

        _plane[i].clear();

        for(j=0; j<n; j++)

```

```

    {
        sita = ((float)j * 2 * pi) / 180.0 * pi;
        u->get_point(xc, yc, sita, num, point, &x, &y);
        _plane[i].add(x, y);
    }
}
delete u;
}

void Cylinder::to_ivFileContour(FILE *fp, float thickness, float pX, float pY, int index)
{
    if(_numFrames <= 1) return;

    fprintf(fp, "Separator {\n");
    fprintf(fp, "    Material {\n");
    fprintf(fp, "        ambientColor      0.595855 0.488367 0.357513\n");

    if(index_obj == 1)
        fprintf(fp, "        diffuseColor      0.9 0.0 0.0\n");
    else if(index_obj == 2)
        fprintf(fp, "        diffuseColor      0.9 0.0 0.9\n");
    else if(index_obj == 3)
        fprintf(fp, "        diffuseColor      0.2 0.9 0.2\n");
    else if(index_obj == 4)
        fprintf(fp, "        diffuseColor      0.0 0.9 0.0\n");
    else if(index_obj == 5)
        fprintf(fp, "        diffuseColor      0.0 0.0 0.9\n");
    else
        fprintf(fp, "        diffuseColor      0.595855 0.488367 0.357513\n");

    fprintf(fp, "        emissiveColor      0 0.7 0\n");
    fprintf(fp, "        shininess          0\n");
    fprintf(fp, "        transparency       0\n");
    fprintf(fp, "    } \n");

    int i, j, i1, i2, i3, i4;
    float x, y, z;

    for(i=0; i<_numFrames; i++)
    {
        fprintf(fp, "\n\n Coordinate3 {\n    point [\n");

        for(j=0; j<_plane[i]._numPoints; j++)
        {
            x = _plane[i]._x[j];
            y = _plane[i]._y[j];
            z = _plane[i]._z[j];
            y = _z[i] * thickness;
            fprintf(fp, "        %f %f %f, \n", x, y, z);
        }
        fprintf(fp, "    ]\n    }\n\n");

        fprintf(fp, "IndexedLineSet {\n        coordIndex [\n");

        for(j=0; j<_plane[i]._numPoints; j++)
            fprintf(fp, "%d, ", j);

        fprintf(fp, "0, -1\n");
        fprintf(fp, "    ]\n    }\n\n");
    }
    fprintf(fp, "}\n\n");
}

void Cylinder::to_ivFileSurface(FILE *fp, float thickness, float pX, float pY, int index)
{
    if(_numFrames <= 1) return;

```

```

fprintf(fp, "Separator");

fprintf(fp, "      Material {\n");

if(index_obj == 1)
{
    fprintf(fp, "      ambientColor      0.9  0.0  0.0\n");
    fprintf(fp, "      diffuseColor      0.9  1.0  0.0\n");
    fprintf(fp, "      specularColor     1.0  0.0  0.0\n");
    fprintf(fp, "      emissiveColor     0.0  0.0  0.0\n");
}
else if(index_obj == 2)
{
    fprintf(fp, "      ambientColor      0.9  0.0  0.0\n");
    fprintf(fp, "      diffuseColor      0.9  1.0  0.0\n");
    fprintf(fp, "      specularColor     1.0  0.0  1.0\n");
    fprintf(fp, "      emissiveColor     0.0  0.0  0.0\n");
}
else if(index_obj == 3)
{
    fprintf(fp, "      ambientColor      0.595855 0.488367 0.357513\n");
    fprintf(fp, "      diffuseColor      0.2  0.9  0.2\n");
    fprintf(fp, "      emissiveColor     0.0  0.7  0.0\n");
}
else
{
    fprintf(fp, "      ambientColor      0.595855 0.488367 0.357513\n");
    fprintf(fp, "      diffuseColor      0.595855 0.488367 0.357513\n");
    fprintf(fp, "      emissiveColor     0.0  0.7  0.0\n");
}

fprintf(fp, "      shininess  0\n");
fprintf(fp, "      transparency  0\n");
fprintf(fp, "    } \n");

int    i, j;
float  x, y, z;

//for(i=0; i<_numFrames; i++)
//  _plane[i].fill();

fprintf(fp, "\n\n Coordinate3 {\n  point [\n");
for(i=0; i<_numFrames; i++)
{
    for(j=0; j<_plane[i]._numPoints; j++)
    {
        
            //x = _x[i] * thickness;
            //y = _y[i] * thickness;
            //z = _z[i] * thickness;
            //fprintf(fp, "      %f %f %f, \n", x, y, z);
        
        y = _z[i] * thickness;
        fprintf(fp, "      %f %f %f, \n", x, y, z);
    }
}
fprintf(fp, "    ]\n  }\n\n");

fprintf(fp, " IndexedFaceSet {\n      coordIndex [\n");

i = 0;
for(j=0; j<_plane[i]._numPoints; j++)
    fprintf(fp, "%d,", j);
fprintf(fp, "0, -1, \n");

for(i=0; i<(_numFrames-1); i++)
{

```

```

    oneLayer(fp, i, _plane[i]._numPoints);
}

i = _numFrames - 1;
for(j=0; j<_plane[i]._numPoints; j++)
    fprintf(fp, "%d,", i*_plane[i]._numPoints + j);
fprintf(fp, "%d, -1, \n", i*_plane[i]._numPoints);

fprintf(fp, "    ]\n    }\n");
fprintf(fp, ")\n\n");
}

void Cylinder::oneLayer(FILE *fp, int i, int num)
{
    int j, i1, i2, i3, i4;

    for(j=0; j<(num-1); j++)
    {
        i1 = i*num + j;
        i2 = i1 + 1;
        i3 = (i+1)*num + j;
        i4 = i3 + 1;
        fprintf(fp, "        %d,%d,%d,%d,    -1, \n", i1, i2, i4, i3);
    }

    i1 = i*num + (num - 1);
    i2 = i*num;
    i3 = (i+1)*num + (num - 1);
    i4 = (i+1)*num;
    fprintf(fp, "        %d,%d,%d,%d,    -1, \n", i1, i2, i4, i3);
}

```

```

////////////////////////////////////
//
// Source file for Win3DMainWindow
//
// This class is a subclass of VkSimpleWindow
//
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Try to restrict any changes to the bodies of functions
// corresponding to menu items, the constructor and destructor.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
// Avoid gratuitous reformatting and other changes that might
// make it difficult to integrate changes made using RapidApp
////////////////////////////////////
#include "Win3DMainWindow.h"

#include <Vk/VkApp.h>
#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include <Inventor/Xt/viewers/SoXtExaminerViewer.h>
#include <Inventor/So.h> // Includes ALL Inventor headers.
                        // Replace for efficiency and faster compilation
#include <Inventor/Xt/viewers/SoXtExaminerViewer.h>

extern void VkUnimplemented ( Widget, const char * );

//--- Start editable code block: headers and declarations

#include "Utility_3D.h"
#include <X11/keysym.h>
#include <Inventor/nodes/SoIndexedFaceSet.h>

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String Win3DMainWindow::_defaultWin3DMainWindowResources[] = {
    "**title: 3D Viewer",

    //--- Start editable code block: Win3DMainWindow Default Resources

    //--- End editable code block: Win3DMainWindow Default Resources

    (char*)NULL
};

```

```

Win3DMainWindow::Win3DMainWindow ( const char *name,
                                   ArgList args,
                                   Cardinal argCount) :
                                   VkSimpleWindow ( name, args, argCount )
{
    // Load any class-default resources for this object

    setDefaultResources ( baseWidget(), _defaultWin3DMainWindowResources );

    // Create the view component contained by this window

    // Add the component as the main view
    // Inventor components are not really VkComponents,
    // so we have to add them by their widgets

    //---- Start editable code block: Win3DMainWindow constructor

    init();

    //---- End editable code block: Win3DMainWindow constructor

}    // End Constructor

Win3DMainWindow::~Win3DMainWindow()
{
    //---- Start editable code block: Win3DMainWindow destructor

    clear();

    //---- End editable code block: Win3DMainWindow destructor
}    // End destructor

const char *Win3DMainWindow::className()
{
    return ("Win3DMainWindow");
}    // End className()

Boolean Win3DMainWindow::okToQuit()
{
    //---- Start editable code block: Win3DMainWindow okToQuit
    printf("\n okToQuit in Win3DMainWindow\n");
    clear();

    // This member function is called when the user quits by calling
    // theApplication->terminate() or uses the window manager close protocol
    // This function can abort the operation by returning FALSE, or do some.
    // cleanup before returning TRUE. The actual decision is normally passed on
    // to the view object

    // The view object is an Inventor component, which does
    // not currently support the okToQuit protocol

    return ( TRUE );

    //---- End editable code block: Win3DMainWindow okToQuit
}    // End okToQuit()

```

```

////////////////////////////////////
// The following functions are called from callbacks
////////////////////////////////////

```

```

//---- Start editable code block: End of generated code

```

```

void Win3DMainWindow::update()
{
    int w, h;
    XWindowAttributes xwa;
    if(XGetWindowAttributes(XtDisplay(mainWindowWidget()), XtWindow(mainWindowWidget()))
    {
        w = xwa.width;
        h = xwa.height;
    }

    printf("\n    Win3DMainWindow    %d    %d\n", w, h);

    Utility_3D *u3D = new Utility_3D();

    _viewer = u3D -> create_iv("/usr/people/meide/rapidapp/.cmis32/ROIS.iv", mainWindow
        (SoXtExaminerViewer *)_viewer, 0, 0, w, h);

    delete u3D;
}

void Win3DMainWindow::update_localizer(SoSeparator *root)
{
    int w, h;
    XWindowAttributes xwa;
    if(XGetWindowAttributes(XtDisplay(mainWindowWidget()), XtWindow(mainWindowWidget()))
    {
        w = xwa.width;
        h = xwa.height;
    }

    printf("\n    Win3DMainWindow    %d    %d\n", w, h);

    Utility_3D *u3D = new Utility_3D();

    if(_viewer == NULL)
    {
        _viewer = u3D -> create_localizer_iv("Flow3D.iv", mainWindowWidget(),
            _viewer, 0, 0, w, h, root);

        printf(" Create Viewer in Win3DMainWindow\n");
        _viewer->setEventCallback(&Win3DMainWindow::myAppEventHandler, this);
    }
    else
    {
        _root -> unref();
        _viewer = u3D -> create_localizer_iv("Flow3D.iv", mainWindowWidget(),
            _viewer, 0, 0, w, h, root);
    }

    printf(" before\n");
    _root = (SoSeparator *) _viewer->getSceneGraph();
    printf(" after\n");
    delete u3D;
}

```

```

SbBool
Win3DMainWindow::myAppEventHandler(void *userData, XAnyEvent *anyevent) 188
{
    Win3DMainWindow *obj = (Win3DMainWindow *) userData;
    return obj -> appEventHandler(userData, anyevent);
}

SbBool
Win3DMainWindow::appEventHandler(void *userData, XAnyEvent *anyevent)
{
    //SoXtRenderArea *myRenderArea = (SoXtRenderArea *) userData;
    XButtonEvent *myButtonEvent;
    XMotionEvent *myMotionEvent;
    SbVec3f vec;
    SbBool handled = TRUE;

    /*
    KeySym    key_symbol;
    char      command[100];

    if(anyevent->type == KeyPress)
    {
        XLookupString( (XKeyEvent *) anyevent->xkey, command, sizeof(command),
            &key_symbol, (XComposeStatus *) NULL);
        switch (key_symbol)
        {
            case XK_Shift_L:
                printf(" XK_Shift_L \n");
                break;
            case XK_Control_L:
                printf("  XK_Control_L\n");
                break;
            default:
                break;
        }
    }
    */

    switch (anyevent->type) {

case ButtonPress:
    myButtonEvent = (XButtonEvent *) anyevent;
    _x1 = myButtonEvent->x;
    _y1 = myButtonEvent->y;
    if (myButtonEvent->button == Button1) {
        _button1 = TRUE;
        //myProjectPoint(myRenderArea,
        //    myButtonEvent->x, myButtonEvent->y, vec);
        //myAddPoint(myRenderArea, vec);
    } else if (myButtonEvent->button == Button2) {
        _button2 = TRUE;
        //myTicker->schedule(); // start spinning the camera
    } else if (myButtonEvent->button == Button3) {
        _button3 = TRUE;
        //myClearPoints(myRenderArea); // clear the point set
    }
    break;

case ButtonRelease:
    myButtonEvent = (XButtonEvent *) anyevent;
    if (myButtonEvent->button == Button1) {
        _button1 = FALSE;
    }
    else if (myButtonEvent->button == Button2) {
        _button2 = FALSE;
        //myTicker->unsubscribe(); // stop spinning the camera
    }
    }
}

```



```

    }
    else if (myButtonEvent->button == Button3) {
        _button3 = FALSE;
    }
    break;

case MotionNotify:
    myMotionEvent = (XMotionEvent *) anyevent;
    _x2 = myMotionEvent->x;
    _y2 = myMotionEvent->y;

    if (myMotionEvent->state & Button1Mask) {
        if(_button1) mouse(1, _x1, _y1, _x2, _y2);
        //printf(" %d %d \n", myMotionEvent->x, myMotionEvent->y);
        //myProjectPoint(myRenderArea,
        //                myMotionEvent->x, myMotionEvent->y, vec);
        //myAddPoint(myRenderArea, vec);
    }
    else if (myMotionEvent->state & Button2Mask)
    {
        if(_button2) mouse(2, _x1, _y1, _x2, _y2);
    }
    else if (myMotionEvent->state & Button3Mask)
    {
        if(_button3) mouse(3, _x1, _y1, _x2, _y2);
    }
    _x1 = _x2;
    _y1 = _y2;

    break;

default:
    handled = FALSE;
    break;
}

return handled;
}

void Win3DMainWindow::mouse(int whichMouse, int x1, int y1, int x2, int y2)
{
    SoTransform *myTransform;
    SbRotation   rot;
    SbVec3f      vec3f;
    float        x, y, z, d;

    x = x2 - x1;
    y = y2 - y1;
    d = fsqrt(x*x + y*y);
    if(d == 0) return;

    switch (_whichScene)
    {
        case _LOC_PLANE:
            myTransform = (SoTransform *) ( (SoSeparator *) ((SoSeparator *)
                _root->getChild(2)) -> getChild(0) ) ->getChild(0);
            act_transform(whichMouse, myTransform, x, y, d);
            break;
        case _LOC_VESSELS:
            myTransform = (SoTransform *) ( (SoSeparator *) ((SoSeparator *)
                _root->getChild(2)) -> getChild(1) ) ->getChild(0);
            act_transform(whichMouse, myTransform, x, y, d);
            break;
        case _LOC_UNIVERSE:
            myTransform = (SoTransform *) _root->getChild(1);
            act_transform(whichMouse, myTransform, x, y, d);

```

```

/*
myTransform = (SoTransform *) ( (SoSeparator *) _root->getChildren(2) -> getChildren(0) ); 190
_act_transform(whichMouse, myTransform, x, y, d);
myTransform = (SoTransform *) ( (SoSeparator *) _root->getChildren(2) -> getChildren(1) );
_act_transform(whichMouse, myTransform, x, y, d);
*/
break;
default:
break;
}
}

void Win3DMainWindow::update_plane()
{
    SoSeparator *obj = (SoSeparator *) (_root->getChildren(2));

    SoSeparator *plane = (SoSeparator *) (obj -> getChildren(4));
    SoSeparator *redBall = (SoSeparator *) (obj -> getChildren(5));
    SoSeparator *yellowBall = (SoSeparator *) (obj -> getChildren(6));

    SoCoordinate3 *pCoord = (SoCoordinate3 *) plane -> getChildren(1);
    SoIndexedFaceSet *pFace = (SoIndexedFaceSet *) plane -> getChildren(2);
    pCoord->point.setValues(0, 4, _planeVertex);
    pFace->coordIndex.setValues(0, 5, _FaceIndex);

    //
    // set up Red Ball
    //
    SoTransform *ballTransform = (SoTransform *) redBall->getChildren(0);
    SoSphere *ballSphere = (SoSphere *) redBall->getChildren(2);

    float xc = _planeVertex[0][0];
    float yc = _planeVertex[0][1];
    float zc = _planeVertex[0][2];

    ballTransform->translation.setValue(xc, yc, zc);
    ballTransform->center.setValue(xc, yc, zc);
    ballSphere -> radius.setValue(2.0);

    //
    // set up Yellow Ball
    //
    SoTransform *ballTransform1 = (SoTransform *) yellowBall->getChildren(0);
    SoSphere *ballSphere1 = (SoSphere *) yellowBall->getChildren(2);

    xc = _planeVertex[1][0];
    yc = _planeVertex[1][1];
    zc = _planeVertex[1][2];

    ballTransform1->translation.setValue(xc, yc, zc);
    ballTransform1->center.setValue(xc, yc, zc);
    ballSphere1 -> radius.setValue(2.0);
}

void Win3DMainWindow::act_transform(int whichMouse, SoTransform *myTransform,
float x, float y, float d)
{
    SbRotation rot;
    SbVec3f vec3f;
    float z;

    switch (whichMouse)
    {

```

```

case 1:
    rot = SbRotation(Vec3f(y/d, x/d, 0), d * M_PI/180.0);      191
    myTransform->rotation.setValue(myTransform->rotation.getValue() * rot);
    break;
case 2:
    vec3f.setValue(x, -y, 0);
    myTransform->translation.setValue(myTransform->translation.getValue() + vec3f);
    break;
case 3:
    vec3f.setValue( y/20.0, y/20.0, y/20.0 );
    vec3f += myTransform->scaleFactor.getValue();
    vec3f.setValue(x, y, z);
    if(x < 0) x = 0;
    if(y < 0) y = 0;
    if(z < 0) z = 0;
    myTransform->scaleFactor.setValue(x, y, z);
    break;
default:
    break;
}
}

```

```
void Win3DMainWindow::init()
```

```

{
    _viewer = NULL;
    _whichScene = _LOC_UNIVERSE;
    _button1 = FALSE;
    _button2 = FALSE;
    _button3 = FALSE;
}

```

```
void Win3DMainWindow::clear()
```

```

{
    printf("\n\n clear Win3DMainWindow\n\n");

    if(_viewer != NULL) delete _viewer;
    _objMag -> _win3D = NULL;
}

```

```
//----- End editable code block: End of generated code
```

User: meide
Host: phoenix
Class: phoenix
Job: ProgressMainWindow.C

192

```

////////////////////////////////////
//
// Source file for Bb
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "Bb.h"
#include <Vk/VkEZ.h>
#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/Separator.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

```

```

// Externally defined classes referenced by this class:

```

```

#include "DeckLTabbedDeck.h"
#include "DeckRTabbedDeck.h"
#include <Vk/VkWindow.h>
#include <Vk/VkMenuBar.h>
#include <Vk/VkSubMenu.h>

```

```

extern void VkUnimplemented ( Widget, const char * );

```

```

////////////////////////////////////
// The following non-container elements are created by BbUI and are
// available as protected data members inherited by this class
//
// VkOptionMenu *          _optionMenuPCMRA
// VkMenuItem *            _optionMagnitude
// VkMenuItem *            _optionPhase
// VkMenuItem *            _optionVelocity
// VkOptionMenu *          _optionMenuAnimate
// VkMenuItem *            _optionAnimate
// VkMenuItem *            _optionStopAnimate
// VkMenuItem *            _optionNewAnimate
// VkOptionMenu *          _optionMenuSpace
// VkMenuItem *            _optionGray2D
// VkMenuItem *            _optionColor2D
// VkMenuItem *            _option3D
// VkOptionMenu *          _optionMenuVisual
// VkMenuItem *            _optionSpline
// VkMenuItem *            _optionSimple
// VkOptionMenu *          _optionMenuSelect

```

```

// VkMenuItem *           _optionWhole
// VkMenuItem *           _optionROI
// VkMenuItem *           _optionReference
// VkMenuItem *           _optionOther
// XmArrowButton          _arrowNext
// XmArrowButton          _arrowPrev
// XmLabel                 _labelImgNumber
// XmSeparator             _separatorTop
// XmSeparator             _separatorBottom
// XmSeparator             _separatorMiddle
//
// The following components are created by BbUI and are
// available as protected data members inherited by this class
//
// DeckRTabbedDeck         *_deckR
// DeckLTabbedDeck         *_deckL
//
////////////////////////////////////

//---- Start editable code block: headers and declarations

#include "Animate.h"
#include "Utility.h"
#include <Vk/VkDeck.h>
#include <time.h>

#include "GE.h"
#include "InfoMainWindow.h"
#include "BbAnimation.h"
#include "BbVisual.h"

//---- End editable code block: headers and declarations

//---- Bb Constructor

Bb::Bb(const char *name, Widget parent) :
    BbUI(name, parent)
{
    // This constructor calls BbUI(parent, name)
    // which calls BbUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: Bb constructor

    //---- End editable code block: Bb constructor

}    // End Constructor

Bb::Bb(const char *name) :
    BbUI(name)
{
    // This constructor calls BbUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: Bb constructor 2

```

```
//---- End editable code block: Bb constructor 2
```

```
} // End Constructor
```

```
Bb::~Bb()
```

```
{
```

```
    // The base class destructors are responsible for  
    // destroying all widgets and objects used in this component.  
    // Only additional items created directly in this class  
    // need to be freed here.
```

```
//---- Start editable code block: Bb destructor
```

```
//---- End editable code block: Bb destructor
```

```
} // End Destructor
```

```
const char * Bb::className() // classname
```

```
{
```

```
    return ("Bb");
```

```
} // End className()
```

```
void Bb::Next ( Widget w, XtPointer callData )
```

```
{
```

```
//---- Start editable code block: Bb Next
```

```
XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;
```

```
//--- Comment out the following line when Bb::Next is implemented:
```

```
//::VkUnimplemented ( w, "Bb::Next" );
```

```
int img_number = _objMag -> msgsLeft.img_number;  
_objMag -> msgsRight.img_number_prev = img_number;
```

```
++img_number;
```

```
if(img_number > _objMag-> msgsLoaded.img_end)  
    img_number = _objMag->msgsLoaded.img_start;
```

```
_objMag->update_Aimg(img_number);
```

```
//---- End editable code block: Bb Next
```

```
} // End Bb::Next()
```

```
void Bb::Prev ( Widget w, XtPointer callData )
```

```
{
```

```
//---- Start editable code block: Bb Prev
```

```
XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;
```

```
//--- Comment out the following line when Bb::Prev is implemented:
```

```

//::VkUnimplemented ( "Bb::Prev" );
int img_number = _objMag->msgsLeft.img_number;
_objMag->msgsRight.img_number_prev = img_number;
--img_number;

if(img_number < _objMag->msgsLoaded.img_start)
    img_number = _objMag->msgsLoaded.img_end;

_objMag->update_Aimg(img_number);

//---- End editable code block: Bb Prev
} // End Bb::Prev()

void Bb::doOption3D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOption3D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOption3D is implemented:
    //::VkUnimplemented ( w, "Bb::doOption3D" );

    _objMag->msgsRight.img_space = IMAGE_3D;
    _objMag->msgsRight.animate_mode = ANIMATE_3D;
    ((BbAnimation *) (_objMag->_RAnimate))->set_toggle(3);
    _objMag->update_RimgView3D();

    //---- End editable code block: Bb doOption3D
} // End Bb::doOption3D()

void Bb::doOptionAnimate ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionAnimate

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionAnimate is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionAnimate" );

    if(_animate != NULL)
    {
        _animate->_time_out = 1;
        animation();
    }

    //---- End editable code block: Bb doOptionAnimate
} // End Bb::doOptionAnimate()

void Bb::doOptionColor2D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionColor2D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionColor2D is implemented:

```



```

//::VkUnimplemented "Bb::doOptionColor2D" );

_objMag->msgsRight.img_space = IMAGE_2D;
_objMag -> msgsRight.animate_mode = ANIMATE_2D;
((BbAnimation *)(_objMag -> _RAnimate)) -> set_toggle(2);
_objMag -> msgsRight.img_visual_type = VISUAL_COLOR;
_objMag -> update_Rvisual(int(VISUAL_COLOR));

//---- End editable code block: Bb doOptionColor2D

} // End Bb::doOptionColor2D()

void Bb::doOptionGray2D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionGray2D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionGray2D is implemented:

    //::VkUnimplemented ( w, "Bb::doOptionGray2D" );

    _objMag->msgsRight.img_space = IMAGE_2D;
    _objMag -> msgsRight.animate_mode = ANIMATE_2D;
    ((BbAnimation *)(_objMag -> _RAnimate)) -> set_toggle(2);
    _objMag -> msgsRight.img_visual_type = VISUAL_GRAY;
    _objMag -> update_Rvisual(int(VISUAL_GRAY));

    //---- End editable code block: Bb doOptionGray2D

} // End Bb::doOptionGray2D()

void Bb::doOptionMagnitude ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionMagnitude

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionMagnitude is implemented:

    //::VkUnimplemented ( w, "Bb::doOptionMagnitude" );

    if(_objMag -> msgsLeft.img_type == IMAGE_PCMRA)
    {
        _objMag -> msgsLeft.img_pcmra_type = PCMRA_MAGNITUDE;
        _objMag -> msgsRight.flowDir = 0;
        if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
            _objMag -> update_Limg( _objMag -> msgsLeft.img_number );
        else
            _objMag -> update_Aimg( _objMag -> msgsLeft.img_number );
    }

    //---- End editable code block: Bb doOptionMagnitude

} // End Bb::doOptionMagnitude()

void Bb::doOptionNewAnimate ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionNewAnimate

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

```

```

//--- Comment out the following line when Bb::doOptionNewAnimate is implemented:
//::VkUnimplemented ( w, "Bb::doOptionNewAnimate" );

_objMag -> create_animate();

//---- End editable code block: Bb doOptionNewAnimate
} // End Bb::doOptionNewAnimate()

void Bb::doOptionOther ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionOther
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when Bb::doOptionOther is implemented:
    ::VkUnimplemented ( w, "Bb::doOptionOther" );

    //---- End editable code block: Bb doOptionOther
} // End Bb::doOptionOther()

void Bb::doOptionPhase ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionPhase
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when Bb::doOptionPhase is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionPhase" );

    if(_objMag -> msgsLeft.img_type == IMAGE_PCMRA)
    {
        _objMag -> msgsLeft.img_pcmra_type = PCMRA_PHASE;
        _objMag -> msgsRight.flowDir = _objMag -> msgsRight.flowDir2;
        if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
            _objMag -> update_Limg( _objMag -> msgsLeft.img_number );
        else
            _objMag -> update_Aimg(_objMag -> msgsLeft.img_number);
    }

    //---- End editable code block: Bb doOptionPhase
} // End Bb::doOptionPhase()

void Bb::doOptionROI ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionROI
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when Bb::doOptionROI is implemented:

```

```

//::VkUnimplemented ( "Bb::doOptionROI" );

if(_objMag -> msgsLeft.user == USER_NOVIES)
{
    _objMag -> msgsRight.img_visual_type = VISUAL_COLOR;
    _objMag -> msgsRight.lowMagColorROI = 38;
    _objMag -> msgsRight.highMagColorROI = 388;

    int roi_type = ROI_ELLIPSE;
    int roi_action = ROI_MODIFY;

    _objMag-> _imgView -> _roi_type = roi_type;
    _objMag-> _imgView -> _roi_action = roi_action;
    _objMag->msgsLeft.roi_type = roi_type;
    _objMag->msgsLeft.roi_action = roi_action;

    _objMag-> _imgView2 -> _roi_type = roi_type;
    _objMag-> _imgView2 -> _roi_action = roi_action;
    _objMag->msgsRight.roi_type = roi_type;
    _objMag->msgsRight.roi_action = roi_action;
}

if(_objMag -> _imgView -> _ROI != NULL && _objMag -> _imgView -> _ROI -> _draw_sta
{
    _objMag -> msgsRight.img_select = RIGHT_IMG_ROI;
    if(_objMag -> _imgView2 -> _ROI != NULL)
    {
        delete _objMag -> _imgView2 -> _ROI;
        _objMag -> _imgView2 -> _ROI = NULL;
    }
    _objMag -> update_Rimg(_objMag -> msgsRight.img_number);

    //( (VkDeck *) ((DeckLTabbedDeck *) (_objMag -> _deckL)) )
    // -> pop( ((VkComponent *) ((BbLROI *) (_objMag -> _LROI))), VkDeck::POP);
}

//---- End editable code block: Bb doOptionROI
} // End Bb::doOptionROI()

void Bb::doOptionReference ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb doOptionReference

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionReference is implemented:

    //::VkUnimplemented ( w, "Bb::doOptionReference" );

    _objMag -> msgsRight.img_select = RIGHT_IMG_REF;
    if(_objMag -> _imgView2 -> _ROI != NULL)
    {
        delete _objMag -> _imgView2 -> _ROI;
        _objMag -> _imgView2 -> _ROI = NULL;
    }
    _objMag -> update_Rimg(_objMag -> msgsRight.img_number);

    //---- End editable code block: Bb doOptionReference
} // End Bb::doOptionReference()

```

```

void Bb::doOptionSimple ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb doOptionSimple

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionSimple is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionSimple" );

    _objMag -> msgsRight.img_scale_type = SCALE_SIMPLE;
    _objMag -> update_RingView(int(SCALE_SIMPLE));

    //----- End editable code block: Bb doOptionSimple
}    // End Bb::doOptionSimple()

void Bb::doOptionSpline ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb doOptionSpline

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionSpline is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionSpline" );

    _objMag -> msgsRight.img_scale_type = SCALE_SPLINE;
    _objMag -> update_RingView(int(SCALE_SPLINE));

    //----- End editable code block: Bb doOptionSpline
}    // End Bb::doOptionSpline()

void Bb::doOptionStopAnimate ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb doOptionStopAnimate

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionStopAnimate is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionStopAnimate" );

    if(_animate != NULL)
        _animate->_time_out = 0;

    //----- End editable code block: Bb doOptionStopAnimate
}    // End Bb::doOptionStopAnimate()

void Bb::doOptionVelocity ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb doOptionVelocity

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionVelocity is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionVelocity" );

```

```

if(_objMag -> msgsLeft.img_type == IMAGE_PCMRA)
{
    _objMag -> msgsLeft.img_pcmra_type = PCMRA_VELOCITY;
    _objMag -> msgsRight.flowDir = _objMag -> msgsRight.flowDir2;
    if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
        _objMag -> update_Limg( _objMag -> msgsLeft.img_number );
    else
        _objMag -> update_Aimg(_objMag -> msgsLeft.img_number);
}

//----- End editable code block: Bb doOptionVelocity
} // End Bb::doOptionVelocity()

void Bb::doOptionWhole ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb doOptionWhole

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb::doOptionWhole is implemented:
    //::VkUnimplemented ( w, "Bb::doOptionWhole" );

    _objMag -> msgsRight.img_select = RIGHT_IMG_WHOLE;
    if(_objMag -> _imgView2 -> _ROI != NULL)
    {
        delete _objMag -> _imgView2 -> _ROI;
        _objMag -> _imgView2 -> _ROI = NULL;
    }
    _objMag -> update_Rimg(_objMag -> msgsRight.img_number);

    //----- End editable code block: Bb doOptionWhole
} // End Bb::doOptionWhole()

void Bb::copy( )
{
    // This member function is called when the user has selected
    // the Copy command from the Edit menu. You should copy the
    // selected object in your program to on the clipboard.

    //----- Start editable code block: Bb copy

    //--- Comment out this line when this function is implemented:
    //::VkUnimplemented ( NULL, "Bb::copy" );

    //----- End editable code block: Bb copy
} // End Bb::copy()

void Bb::cut( )
{
    // This member function is called when the user has selected

```

```
// the Cut command from the Edit menu. You should cut the  
// selected object in your program and place it on the clipboard.
```

201

```
//---- Start editable code block: Bb cut  
//--- Comment out this line when this function is implemented:  
::VkUnimplemented ( NULL, "Bb::cut" );  
  
//---- End editable code block: Bb cut  
} // End Bb::cut()  
  
void Bb::expertCallback( Widget w, XtPointer callData )  
{  
  
//---- Start editable code block: Bb expertCallback  
//--- Comment out this line when this function is implemented:  
//::VkUnimplemented ( NULL, "Bb::expertCallback" );  
_objMag -> msgsLeft.user = USER_EXPERT;  
  
//---- End editable code block: Bb expertCallback  
} // End Bb::expertCallback()  
  
void Bb::imgInfoCallback( Widget w, XtPointer callData )  
{  
  
//---- Start editable code block: Bb imgInfoCallback  
//--- Comment out this line when this function is implemented:  
::VkUnimplemented ( NULL, "Bb::imgInfoCallback" );  
  
//---- End editable code block: Bb imgInfoCallback  
} // End Bb::imgInfoCallback()  
  
void Bb::mraInfoCallback( Widget w, XtPointer callData )  
{  
  
//---- Start editable code block: Bb mraInfoCallback  
//--- Comment out this line when this function is implemented:  
//::VkUnimplemented ( NULL, "Bb::mraInfoCallback" );  
char str1[100], str2[100], str3[100], str4[100], str5[100], str6[100];  
char str7[100], str8[100];
```

```

char str2A[100], str2B[100], str2C[100], str2D[100], str2E[100];
char str[2000];

char strCR[50], strCA[50], strCS[50];

char strTLR[50], strTLA[50], strTLS[50];

GE_PCMRA_HEADER_OBJ *pc = _objMag -> _img -> get_header();

sprintf(str1, "Width:           %d\n", pc->img_width);
sprintf(str2, "Height:           %d\n\n", pc->img_height);

sprintf(str2A, "Dim_X:           %f\n", pc->dim_X);
sprintf(str2B, "Dim_Y:           %f\n", pc->dim_Y);
sprintf(str2C, "ScanSpacing:      %f\n", pc->scanspacing);
sprintf(str2D, "pixsize_X:        %f\n", pc->pixsize_X);
sprintf(str2E, "pixsize_Y:        %f\n\n", pc->pixsize_Y);

sprintf(str3, "Thickness:         %f\n", pc->slthick);
sprintf(str4, "FOV:              %f\n", pc->dfov);
sprintf(str5, "Heart Rate: %d Delay Time: %d\n\n",
    pc->heart_rate, pc->delay_time);

if(pc->heart_rate > 0)
{
    _objMag -> msgsRight.HR = pc->heart_rate;
}

sprintf(str6, "VENC:           %d\n", pc->pc_venc);
sprintf(str7, "Magnitude Mask: %d\n", pc->mag_weighting_flag);
sprintf(str8, "VENC Scale:      %f\n", pc->venc_weighted_scale);

sprintf(strCR, "Center_R:         %6.2f\n", pc->ctr_R);
sprintf(strCA, "Center_A:         %6.2f\n", pc->ctr_A);
sprintf(strCS, "Center_S:         %6.2f\n", pc->ctr_S);

sprintf(strTLR, "Top Left Hand R:      %6.2f\n", pc->tlh_R);
sprintf(strTLA, "Top Left Hand A:      %6.2f\n", pc->tlh_A);
sprintf(strTLS, "Top Left Hand S:      %6.2f\n", pc->tlh_S);

sprintf(str, "%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s\n", str1, str2,
    str2A, str2B, str2C, str2D, str2E, str3, str4,
    str5, str6, str7, str8, strCR, strCA, strCS, strTLR, strTLA, strTLS);

printf("Top Right Hand R:      %6.2f\n", pc->trh_R);
printf("Top Right Hand A:      %6.2f\n", pc->trh_A);
printf("Top Right Hand S:      %6.2f\n", pc->trh_S);

printf("Bottom Right Hand R:    %6.2f\n", pc->brh_R);
printf("Bottom Right Hand A:    %6.2f\n", pc->brh_A);
printf("Bottom Right Hand S:    %6.2f\n", pc->brh_S);

printf("Norm R:           %6.2f\n", pc->norm_R);
printf("Norm A:           %6.2f\n", pc->norm_A);
printf("Norm S:           %6.2f\n", pc->norm_S);

printf("TR:              %d\n", pc->tr);
printf("TE:              %d\n", pc->te);
printf("num_excitations:  %6.2f\n", pc->num_excitations);

InfoMainWindow *info = new InfoMainWindow("info");
info->set(str);
info->show();

//---- End editable code block: Bb mraInfoCallback

```

```
void Bb::newFile( )
{
```

```
//---- Start editable code block: Bb newFile
```

```
//--- Comment out this line when this function is implemented:
```

```
::VkUnimplemented ( NULL, "Bb::newFile" );
```

```
//---- End editable code block: Bb newFile
```

```
} // End Bb::newFile()
```

```
void Bb::noviesCallback( Widget w, XtPointer callData )
{
```

```
//---- Start editable code block: Bb noviesCallback
```

```
//--- Comment out this line when this function is implemented:
```

```
::VkUnimplemented ( NULL, "Bb::noviesCallback" );
```

```
_objMag -> msgsLeft.user = USER_NOVIES;
```

```
//---- End editable code block: Bb noviesCallback
```

```
} // End Bb::noviesCallback()
```

```
void Bb::openFile( const char * filename )
{
```

```
// This member function is called after the user has selected a new
// file to be opened. The name of the file is given by the
// filename argument. You can get additional information by
// examining the state of theFileSelectionDialog object.
```

```
//---- Start editable code block: Bb openFile
```

```
//--- Comment out this line when this function is implemented:
```

```
::VkUnimplemented ( NULL, "Bb::openFile" );
```

```
char  fname[300];
char  str[300];
int   flag, i, num, tmp, vessel;
float heart;
```

```
sprintf(fname, "%s", filename);
```

```
//printf(" %s\n", fname);
```

```
FILE *fp = fopen(fname, "r");
```



```

flag = 0;
do
{
    fscanf(fp, "%s", str);
    if(strcmp(str, "CANVAS_PCMRA_FLOW_97") == 0) ++flag;
    //printf(" %s  %d  %d\n", str, strlen(str), flag);
} while( flag != 1 && !feof(fp));

if(feof(fp)) {fclose(fp); return;}

vessel = _objMag -> _num_vessels;
_objMag -> _vessel = vessel;
++(_objMag -> _num_vessels);

fscanf(fp, "%s", _objMag -> _flow[vessel].vesselName);
fscanf(fp, "%f", &heart);
fscanf(fp, "%d", &num);

printf("  %s  %d\n", _objMag -> _flow[vessel].vesselName, num);
_objMag -> _flow[vessel].numPoints = num;

for(i=0; i<num; i++)
{
    fscanf(fp, "%d %f %f %f %f %f", &tmp, &(_objMag -> _flow[vessel].vesselFlows[i].vf
        &(_objMag -> _flow[vessel].vesselFlows[i].psv), &(_objMag -> _flow[vessel].vesselF
        &(_objMag -> _flow[vessel].vesselFlows[i].mv), &(_objMag -> _flow[vessel].vesselFl
    )
}

fclose(fp);

fp = fopen("clark", "w");
i = 0;
int j;
while(i < num)
{
    for(j=0; j<8; j++)
    {
        fprintf(fp, "%10.5f", _objMag -> _flow[vessel].vesselFlows[i].vfr);
        i++;
    }
    fprintf(fp, "\n");
}
fclose(fp);

((BbVisual *) (_objMag -> _RVis1)) -> add_flow(_objMag -> _flow[vessel].vesselName);

//---- End editable code block: Bb openFile
}    // End Bb::openFile()

void Bb::paste(    )
{
    // This member function is called when the user has selected
    // the Paste command from the Edit menu. You should retrieve
    // the contents of the clipboard and insert it into your
    // program as appropriate.

    //---- Start editable code block: Bb paste

    //--- Comment out this line when this function is implemented:

    ::VkUnimplemented ( NULL, "Bb::paste" );

```

```

//---- End editable code block: Bb paste
}    // End Bb::paste()

void Bb::pcmraCutCallback( Widget w, XtPointer callData )
{

    //---- Start editable code block: Bb pcmraCutCallback
    //--- Comment out this line when this function is implemented:
    ::VkUnimplemented ( NULL, "Bb::pcmraCutCallback" );

    //---- End editable code block: Bb pcmraCutCallback
}    // End Bb::pcmraCutCallback()

void Bb::print( const char * filename )
{

    //---- Start editable code block: Bb print
    //--- Comment out this line when this function is implemented:
    ::VkUnimplemented ( NULL, "Bb::print" );

    //---- End editable code block: Bb print
}    // End Bb::print()

void Bb::save(    )
{
    // This member function is called after the user has selected
    // a file to which to save. The name of the file is given by the
    // filename argument. You can get additional information by
    // examining the state of theFileSelectionDialog object.

    //---- Start editable code block: Bb save
    //--- Comment out this line when this function is implemented:
    ::VkUnimplemented ( NULL, "Bb::save" );

    //---- End editable code block: Bb save
}    // End Bb::save()

```

```

void Bb::saveas( const char* filename )
{
    // This member function is called after the user has selected
    // a file to which to save. The name of the file is given by the
    // filename argument. You can get additional information by
    // examining the state of theFileSelectionDialog object.

    //---- Start editable code block: Bb saveas

    //--- Comment out this line when this function is implemented:

    //::VkUnimplemented ( NULL, "Bb::saveas" );

    char fname[300];
    sprintf(fname, "%s", filename);

    int vessel = _objMag -> _vessel;

    FILE *fp = fopen(fname, "w");

    fprintf(fp, "\n*****\nPatient: \n");
    fprintf(fp, "Anatomy: %s\n", _objMag -> _flow[vessel].vesselName);
    fprintf(fp, "Canvas User: %s\n", _objMag -> _flow[vessel].userName);

    time_t t = time(NULL);
    fprintf(fp, "Date: %s\n", asctime(localtime(&t)));

    GE_PCMRA_HEADER_OBJ *pc = _objMag -> _img -> get_header();
    short heart_rate = pc -> heart_rate;

    fprintf(fp, "\nCANVAS_PCMRA_FLOW_97\n");
    fprintf(fp, "%s\n", _objMag -> _flow[vessel].vesselName);
    fprintf(fp, "%d\n", heart_rate);
    fprintf(fp, "%d\n", _objMag->msgsRight.num_imgs);

    float avg = 0;

    for(int i=0; i<_objMag->msgsRight.num_imgs; i++)
    {
        fprintf(fp, "%d %f %f %f %f %f\n", i+1, _objMag -> _flow[vessel].vesselFlows[i].v,
            _objMag -> _flow[vessel].vesselFlows[i].psv, _objMag -> _flow[vessel].vesselFlows[
            _objMag -> _flow[vessel].vesselFlows[i].mv, _objMag -> _flow[vessel].vesselFlows[i]
            avg += _objMag -> _flow[vessel].vesselFlows[i].vfr;
    }

    avg /= float(_objMag->msgsRight.num_imgs);
    fprintf(fp, "\n\nAverage Flow Rate: %f mL/min\n", avg);

    fclose(fp);

    //---- End editable code block: Bb saveas

} // End Bb::saveas()

```

```

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

```

```

VkComponent *Bb::CreateBb( const char *name, Widget parent )
{
    VkComponent *obj = new Bb ( name, parent );
    return ( obj );
} // End CreateBb

```

```

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

```

```

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *Bb::RegisterBbInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int                (Use XmRInt)
    //     float              (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg")
    //     A filename        (Use VkRFilename or "Filename")
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbInterface()

```

//---- End of generated code

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for Bb3D
//
// This file is generated by RapidApp 1.2
//
// This class is derived from Bb3DUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "Bb3D.h"
#include <Vk/VkEZ.h>
#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

```

```
extern void VkUnimplemented ( Widget, const char * );
```

```

////////////////////////////////////
// The following non-container elements are created by Bb3DUI and are
// available as protected data members inherited by this class
//
// XmTextField          _textfieldHeightFactor
// VkOptionMenu *       _optionMenu2
// VkMenuItem *         _optionSetting3D
// VkMenuItem *         _optionFixed
// XmArrowButton        _arrowHeightDn
// XmArrowButton        _arrowHeightUp
// XmArrowButton        _arrowYPosDn
// XmArrowButton        _arrowYPosUp
// XmTextField          _textfieldHeight3D
// XmTextField          _textfieldYPos
// VkOptionMenu *       _optionMenu1
// VkMenuItem *         _optionOrthoCamera
// VkMenuItem *         _optionPersCamera
// VkMenuItem *         _optionPersCameraRot
// VkOptionMenu *       _optionMenu
// VkMenuItem *         _optionFlowASIS
// VkMenuItem *         _optionFlowReverse
// XmLabel              _labelHigh3D
// XmLabel              _labelLow3D
// XmTextField          _textfieldLow3D
// XmLabel              _labelLow
// XmTextField          _textfieldHigh3D
// XmLabel              _labelHigh
// XmPushButton         _buttonNormalize

```

//
////////////////////////////////////210

//----- Start editable code block: headers and declarations

```
#include "Utility.h"
#include <Vk/VkFormat.h>
```

//----- End editable code block: headers and declarations

//----- Bb3D Constructor

```
Bb3D::Bb3D(const char *name, Widget parent) :
    Bb3DUI(name, parent)
{
    // This constructor calls Bb3DUI(parent, name)
    // which calls Bb3DUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: Bb3D constructor

    //----- End editable code block: Bb3D constructor

}    // End Constructor
```

```
Bb3D::Bb3D(const char *name) :
    Bb3DUI(name)
{
    // This constructor calls Bb3DUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //----- Start editable code block: Bb3D constructor 2

    //----- End editable code block: Bb3D constructor 2

}    // End Constructor
```

```
Bb3D::~Bb3D()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //----- Start editable code block: Bb3D destructor

    //----- End editable code block: Bb3D destructor
```

```
const char * Bb3D::className() // classname
{
    return ("Bb3D");
} // End className()
```

```
void Bb3D::HeightDn ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb3D HeightDn

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::HeightDn is implemented:

    //::VkUnimplemented ( w, "Bb3D::HeightDn" );

    float zoom = atof(XmTextFieldGetString(_textfieldHeightFactor));
    _objMag->msgsRight.Height3D -= 1.0*zoom;
    XmTextFieldSetString(_textfieldHeight3D, (char *)VkFormat("%f",
        _objMag->msgsRight.Height3D));
    update();

    //----- End editable code block: Bb3D HeightDn
} // End Bb3D::HeightDn()
```

```
void Bb3D::HeightUp ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb3D HeightUp

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::HeightUp is implemented:

    //::VkUnimplemented ( w, "Bb3D::HeightUp" );

    float zoom = atof(XmTextFieldGetString(_textfieldHeightFactor));
    _objMag->msgsRight.Height3D += 1.0*zoom;
    XmTextFieldSetString(_textfieldHeight3D, (char *)VkFormat("%f",
        _objMag->msgsRight.Height3D));
    update();

    //----- End editable code block: Bb3D HeightUp
} // End Bb3D::HeightUp()
```

```
void Bb3D::High3D ( Widget w, XtPointer callData )
{
    //----- Start editable code block: Bb3D High3D

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::High3D is implemented:

    ::VkUnimplemented ( w, "Bb3D::High3D" );

    //----- End editable code block: Bb3D High3D
```



```

} // End Bb3D::High3D()

void Bb3D::Low3D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D Low3D

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::Low3D is implemented:
    ::VkUnimplemented ( w, "Bb3D::Low3D" );

    //---- End editable code block: Bb3D Low3D
} // End Bb3D::Low3D()

void Bb3D::TextHeight3D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D TextHeight3D

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::TextHeight3D is implemented:
    //::VkUnimplemented ( w, "Bb3D::TextHeight3D" );

    _objMag->msgsRight.Height3D = atof(XmTextFieldGetString(w));
    update();

    //---- End editable code block: Bb3D TextHeight3D
} // End Bb3D::TextHeight3D()

void Bb3D::TextYPos ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D TextYPos

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::TextYPos is implemented:
    //::VkUnimplemented ( w, "Bb3D::TextYPos" );

    _objMag->msgsRight.YPos3D = atof(XmTextFieldGetString(w));
    update();

    //---- End editable code block: Bb3D TextYPos
} // End Bb3D::TextYPos()

void Bb3D::YPosDn ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D YPosDn

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::YPosDn is implemented:

```

```

//::VkUnimplemented ( w, "Bb3D::YPosDn" );

float zoom = atof(XmTextFieldGetString(_textfieldHeightFactor));
_objMag->msgsRight.YPos3D += 1.0*zoom;
XmTextFieldSetString(_textfieldYPos, (char *)VkFormat("%f",
_objMag->msgsRight.YPos3D));
update();

//---- End editable code block: Bb3D YPosDn
} // End Bb3D::YPosDn()

void Bb3D::YPosUp ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D YPosUp

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::YPosUp is implemented:

    //::VkUnimplemented ( w, "Bb3D::YPosUp" );

    float zoom = atof(XmTextFieldGetString(_textfieldHeightFactor));
    _objMag->msgsRight.YPos3D -= 1.0*zoom;
    XmTextFieldSetString(_textfieldYPos, (char *)VkFormat("%f",
_objMag->msgsRight.YPos3D));
    update();

    //---- End editable code block: Bb3D YPosUp
} // End Bb3D::YPosUp()

void Bb3D::doButtonNormalize ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D doButtonNormalize

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::doButtonNormalize is implemented:

    //::VkUnimplemented ( w, "Bb3D::doButtonNormalize" );

    _objMag->msgsRight.ratio3D = 0;
    update();

    if(_objMag->msgsRight.img_space == IMAGE_3D ||
_objMag->msgsLeft.img_space == IMAGE_3D)
    {
        char str[50];
        sprintf(str, "%5.2f", _objMag->msgsRight.YPos3D);
        XmTextFieldSetString(_textfieldYPos, str);
        sprintf(str, "%5.2f", _objMag->msgsRight.Height3D);
        XmTextFieldSetString(_textfieldHeight3D, str);
    }
    //---- End editable code block: Bb3D doButtonNormalize
} // End Bb3D::doButtonNormalize()

void Bb3D::doOptionFixed ( Widget w, XtPointer callData )
{

```

```

//---- Start editable code block: Bb3D doOptionFixed
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when Bb3D::doOptionFixed is implemented:
//::VkUnimplemented ( w, "Bb3D::doOptionFixed" );

if(_objMag->msgsRight.img_space == IMAGE_3D ||
    _objMag->msgsLeft.img_space == IMAGE_3D)
{
    _objMag->msgsRight.Fixed3D = 1;
}

//---- End editable code block: Bb3D doOptionFixed
} // End Bb3D::doOptionFixed()

void Bb3D::doOptionFlowASIS ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D doOptionFlowASIS
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::doOptionFlowASIS is implemented:
    //::VkUnimplemented ( w, "Bb3D::doOptionFlowASIS" );

    _objMag->msgsRight.flow3DDir = 1;

    //---- End editable code block: Bb3D doOptionFlowASIS
} // End Bb3D::doOptionFlowASIS()

void Bb3D::doOptionFlowReverse ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D doOptionFlowReverse
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::doOptionFlowReverse is implemented:
    //::VkUnimplemented ( w, "Bb3D::doOptionFlowReverse" );

    _objMag->msgsRight.flow3DDir = -1;

    //---- End editable code block: Bb3D doOptionFlowReverse
} // End Bb3D::doOptionFlowReverse()

void Bb3D::doOptionOrthoCamera ( Widget w, XtPointer callData )
{
    //---- Start editable code block: Bb3D doOptionOrthoCamera
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when Bb3D::doOptionOrthoCamera is implemented:
    //::VkUnimplemented ( w, "Bb3D::doOptionOrthoCamera" );

```

```
_objMag->msgsRight.camera = CAMERA_ORTHO;  
update();
```

215

```
//---- End editable code block: Bb3D doOptionOrthoCamera  
}  
// End Bb3D::doOptionOrthoCamera()  
  
void Bb3D::doOptionPersCamera ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: Bb3D doOptionPersCamera  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when Bb3D::doOptionPersCamera is implemented:  
    //::VkUnimplemented ( w, "Bb3D::doOptionPersCamera" );  
  
    _objMag->msgsRight.camera = CAMERA_PERSPECTIVE;  
    update();  
  
    //---- End editable code block: Bb3D doOptionPersCamera  
}  
// End Bb3D::doOptionPersCamera()  
  
void Bb3D::doOptionPersCameraRot ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: Bb3D doOptionPersCameraRot  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when Bb3D::doOptionPersCameraRot is implemented:  
    //::VkUnimplemented ( w, "Bb3D::doOptionPersCameraRot" );  
  
    _objMag->msgsRight.camera = CAMERA_PERSPECTIVE_ROT;  
    update();  
  
    //---- End editable code block: Bb3D doOptionPersCameraRot  
}  
// End Bb3D::doOptionPersCameraRot()  
  
void Bb3D::doOptionSetting3D ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: Bb3D doOptionSetting3D  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when Bb3D::doOptionSetting3D is implemented:  
    //::VkUnimplemented ( w, "Bb3D::doOptionSetting3D" );  
  
    if(_objMag->msgsRight.img_space == IMAGE_3D ||  
        _objMag->msgsLeft.img_space == IMAGE_3D)  
    {  
        _objMag->msgsRight.Fixed3D = 0;  
    }  
}
```

```

//----- End editable code block: Bb3D doOptionSetting3D()

} // End Bb3D::doOptionSetting3D()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *Bb3D::CreateBb3D( const char *name, Widget parent )
{
    VkComponent *obj = new Bb3D ( name, parent );
    return ( obj );
} // End CreateBb3D

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *Bb3D::RegisterBb3DInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char * (Use XmRString)
    // Boolean (Use XmRBoolean)
    // int (Use XmRInt)
    // float (Use XmRFloat)
    // No argument (Use VkrNoArg or "NoArg")
    // A filename (Use VkrFilename or "Filename")
    // An enumeration (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    // A callback (Use XmRCallback)

    static InterfaceMap map[] = {
//----- Start editable code block: Bb3DUI resource table

```

```
    // { "resourceName" "setAttribute", XmRString},  
    //---- End editable code block: Bb3DUI resource table  
    { NULL }, // MUST be NULL terminated  
};  
  
    return map;  
} // End RegisterBb3DInterface()
```

```
//---- End of generated code
```

```
//---- Start editable code block: End of generated code
```

```
void Bb3D::update()  
{  
    if(_objMag->msgsRight.img_space == IMAGE_3D)  
        _objMag->update_RimgView3D();  
  
    if(_objMag->msgsLeft.img_space == IMAGE_3D)  
        _objMag->update_LimgView3D();  
}
```

```
//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for Bb3DUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "Bb3DUI.h" // Generated header file for this class

#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
/*--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String Bb3DUI::_defaultBb3DUIResources[] = {
    "buttonNormalize.labelString: Normalize",
    "labelHigh.labelString: High",
    "labelHigh3D.labelString: 1",
    "labelLow.labelString: Low",
    "labelLow3D.labelString: -1",
    "optionFixed.labelString: Fixed",
    "optionFlowASIS.labelString: Flow AS IS",
    "optionFlowReverse.labelString: Flow Reverse",
    "optionOrthoCamera.labelString: Ortho",
    "optionPersCamera.labelString: Perspective",
    "optionPersCameraRot.labelString: Perspective Rot",
    "optionSetting3D.labelString: Setting",
    "tabLabel: 3D",
    "textfieldHeightFactor.value: 1",

    /*--- Start editable code block: Bb3DUI Default Resources

    //--- End editable code block: Bb3DUI Default Resources

```

};

Bb3DUI::Bb3DUI (const char *name) : VkComponent (name)

```
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //----- Start editable code block: Bb3D constructor 2

    //----- End editable code block: Bb3D constructor 2

}    // End Constructor
```

Bb3DUI::Bb3DUI (const char *name, Widget parent) : VkComponent (name)

```
{
    //----- Start editable code block: Bb3D pre-create

    //----- End editable code block: Bb3D pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //----- Start editable code block: Bb3D constructor

    //----- End editable code block: Bb3D constructor

}    // End Constructor
```

Bb3DUI::~Bb3DUI()

```
{
    // Base class destroys widgets

    //----- Start editable code block: Bb3DUI destructor

    //----- End editable code block: Bb3DUI destructor
}    // End destructor
```

void Bb3DUI::create (Widget parent)

```
{
    Arg      args[8];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBb3DUIResources );
```


// Create an unmanaged widget as the top of the widget hierarchy 220

```
_baseWidget = _bb3D = XtVaCreateWidget ( _name,  
                                         xmBulletinBoardWidgetClass,  
                                         parent,  
                                         XmNresizePolicy, XmRESIZE_GROW,  
                                         (XtPointer) NULL );
```

// install a callback to guard against unexpected widget destruction

```
installDestroyHandler();
```

// Create widgets used in this component
// All variables are data members of this class

```
_textfieldHeightFactor = XtVaCreateManagedWidget ( "textfieldHeightFactor",  
                                                     xmTextFieldWidgetClass,  
                                                     _baseWidget,  
                                                     XmNcolumns, 7,  
                                                     XmNx, 82,  
                                                     XmNy, 127,  
                                                     XmNheight, 35,  
                                                     (XtPointer) NULL );
```

```
_optionMenu2 = new VkOptionMenu ( _baseWidget, "optionMenu2");  
_optionSetting3D = _optionMenu2->addAction ( "optionSetting3D",  
                                              &Bb3DUI::doOptionSetting3DCallback,  
                                              (XtPointer) this );
```

```
_optionFixed = _optionMenu2->addAction ( "optionFixed",  
                                          &Bb3DUI::doOptionFixedCallback,  
                                          (XtPointer) this );
```

```
_arrowHeightDn = XtVaCreateManagedWidget ( "arrowHeightDn",  
                                             xmArrowButtonWidgetClass,  
                                             _baseWidget,  
                                             XmNarrowDirection, XmARROW_DOWN,  
                                             XmNx, 131,  
                                             XmNy, 86,  
                                             XmNwidth, 60,  
                                             XmNheight, 40,  
                                             (XtPointer) NULL );
```

```
XtAddCallback ( _arrowHeightDn,  
               XmNactivateCallback,  
               &Bb3DUI::HeightDnCallback,  
               (XtPointer) this );
```

```
_arrowHeightUp = XtVaCreateManagedWidget ( "arrowHeightUp",  
                                             xmArrowButtonWidgetClass,  
                                             _baseWidget,  
                                             XmNx, 130,  
                                             XmNy, 10,  
                                             XmNwidth, 60,  
                                             XmNheight, 40,  
                                             (XtPointer) NULL );
```

```
XtAddCallback ( _arrowHeightUp,  
               XmNactivateCallback,  
               &Bb3DUI::HeightUpCallback,  
               (XtPointer) this );
```

```

_arrowYPosDn = XtVaCreateManagedWidget ( "arrowYPosDn",
    xmArrowButtonWidgetClass,
    _baseWidget,
    XmNarrowDirection, XmARROW_DOWN,
    XmNx, 40,
    XmNy, 85,
    XmNwidth, 60,
    XmNheight, 40,
    (XtPointer) NULL );

XtAddCallback ( _arrowYPosDn,
    XmNactivateCallback,
    &Bb3DUI::YPosDnCallback,
    (XtPointer) this );

_arrowYPosUp = XtVaCreateManagedWidget ( "arrowYPosUp",
    xmArrowButtonWidgetClass,
    _baseWidget,
    XmNx, 39,
    XmNy, 10,
    XmNwidth, 60,
    XmNheight, 40,
    (XtPointer) NULL );

XtAddCallback ( _arrowYPosUp,
    XmNactivateCallback,
    &Bb3DUI::YPosUpCallback,
    (XtPointer) this );

_textfieldHeight3D = XtVaCreateManagedWidget ( "textfieldHeight3D",
    xmTextFieldWidgetClass,
    _baseWidget,
    XmNcolumns, 5,
    XmNx, 130,
    XmNy, 51,
    XmNheight, 35,
    (XtPointer) NULL );

XtAddCallback ( _textfieldHeight3D,
    XmNactivateCallback,
    &Bb3DUI::TextHeight3DCallback,
    (XtPointer) this );

_textfieldYPos = XtVaCreateManagedWidget ( "textfieldYPos",
    xmTextFieldWidgetClass,
    _baseWidget,
    XmNcolumns, 5,
    XmNx, 38,
    XmNy, 50,
    XmNheight, 35,
    (XtPointer) NULL );

XtAddCallback ( _textfieldYPos,
    XmNactivateCallback,
    &Bb3DUI::TextYPosCallback,
    (XtPointer) this );

_optionMenu1 = new VkOptionMenu ( _baseWidget, "optionMenu1");
_optionOrthoCamera = _optionMenu1->addAction ( "optionOrthoCamera",
    &Bb3DUI::doOptionOrthoCameraCallback,
    (XtPointer) this );

```

```

_optionPersCamera = _optionMenu1->addAction ( "optionPersCamera", 222
&Bb3DUI::doOptionPersCameraCallback,
(XtPointer) this );

_optionPersCameraRot = _optionMenu1->addAction ( "optionPersCameraRot",
&Bb3DUI::doOptionPersCameraRotCal
(XtPointer) this );

_optionMenu = new VkOptionMenu ( _baseWidget, "optionMenu");
_optionFlowASIS = _optionMenu->addAction ( "optionFlowASIS",
&Bb3DUI::doOptionFlowASISCallback,
(XtPointer) this );

_optionFlowReverse = _optionMenu->addAction ( "optionFlowReverse",
&Bb3DUI::doOptionFlowReverseCallback
(XtPointer) this );

_labelHigh3D = XtVaCreateManagedWidget ( "labelHigh3D",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 385,
XmNy, 58,
XmNwidth, 20,
XmNheight, 20,
(XtPointer) NULL );

_labelLow3D = XtVaCreateManagedWidget ( "labelLow3D",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 379,
XmNy, 18,
XmNwidth, 20,
XmNheight, 20,
(XtPointer) NULL );

_textfieldLow3D = XtVaCreateManagedWidget ( "textfieldLow3D",
xmTextFieldWidgetClass,
_baseWidget,
XmNcolumns, 7,
XmNx, 280,
XmNy, 10,
XmNheight, 35,
(XtPointer) NULL );

XtAddCallback ( _textfieldLow3D,
XmNactivateCallback,
&Bb3DUI::Low3DCallback,
(XtPointer) this );

_labelLow = XtVaCreateManagedWidget ( "labelLow",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 230,
XmNy, 18,
XmNwidth, 40,
XmNheight, 20,
(XtPointer) NULL );

```

```

_textfieldHigh3D = XtVaCreateManagedWidget ( "textfieldHigh3D",
                                              xmTextWidgetClass, 223
                                              _baseWidget,
                                              XmNcolumns, 7,
                                              XmNx, 280,
                                              XmNy, 50,
                                              XmNheight, 35,
                                              (XtPointer) NULL );

```

```

XtAddCallback ( _textfieldHigh3D,
                XmNactivateCallback,
                &Bb3DUI::High3DCallback,
                (XtPointer) this );

```

```

_labelHigh = XtVaCreateManagedWidget ( "labelHigh",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 232,
                                         XmNy, 58,
                                         XmNwidth, 36,
                                         XmNheight, 20,
                                         (XtPointer) NULL );

```

```

_buttonNormalize = XtVaCreateManagedWidget ( "buttonNormalize",
                                              xmPushButtonWidgetClass,
                                              _baseWidget,
                                              XmNlabelType, XmSTRING,
                                              XmNx, 260,
                                              XmNy, 90,
                                              XmNwidth, 100,
                                              XmNheight, 40,
                                              (XtPointer) NULL );

```

```

XtAddCallback ( _buttonNormalize,
                XmNactivateCallback,
                &Bb3DUI::doButtonNormalizeCallback,
                (XtPointer) this );

```

```

XtVaSetValues ( _optionMenu2->baseWidget(),
                XmNx, 471,
                XmNy, 10,
                XmNwidth, 111,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

XtVaSetValues ( _optionMenu1->baseWidget(),
                XmNx, 411,
                XmNy, 100,
                XmNwidth, 171,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

XtVaSetValues ( _optionMenu->baseWidget(),
                XmNx, 422,
                XmNy, 55,
                XmNwidth, 156,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

//---- Start editable code block: Bb3DUI create

```

```

//---- End editable code block: Bb3DUI create

```

```

}

```

```

const char * Bb3DUI::className()
{
    return ("Bb3DUI");
} // End className()

```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void Bb3DUI::HeightDnCallback ( Widget      w,
                               XtPointer clientData,
                               XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->HeightDn ( w, callData );
}

```

```

void Bb3DUI::HeightUpCallback ( Widget      w,
                               XtPointer clientData,
                               XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->HeightUp ( w, callData );
}

```

```

void Bb3DUI::High3DCallback ( Widget      w,
                              XtPointer clientData,
                              XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->High3D ( w, callData );
}

```

```

void Bb3DUI::Low3DCallback ( Widget      w,
                             XtPointer clientData,
                             XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->Low3D ( w, callData );
}

```

```

void Bb3DUI::TextHeight3DCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->TextHeight3D ( w, callData );
}

```

```

void Bb3DUI::TextYPosCallback ( Widget      w,
                               XtPointer clientData,
                               XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->TextYPos ( w, callData );
}

```

```

void Bb3DUI::YPosDnCallback ( Widget      w,
                              XtPointer clientData,
                              XtPointer callData )
{

```

```
Bb3DUI* obj = ( Bb3DUI * ) clientData;  
obj->YPosDn ( w, callData );
```

225

```
void Bb3DUI::YPosUpCallback ( Widget      w,  
                             XtPointer clientData,  
                             XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->YPosUp ( w, callData );  
}
```

```
void Bb3DUI::doButtonNormalizeCallback ( Widget      w,  
                                       XtPointer clientData,  
                                       XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doButtonNormalize ( w, callData );  
}
```

```
void Bb3DUI::doOptionFixedCallback ( Widget      w,  
                                    XtPointer clientData,  
                                    XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionFixed ( w, callData );  
}
```

```
void Bb3DUI::doOptionFlowASISCallback ( Widget      w,  
                                       XtPointer clientData,  
                                       XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionFlowASIS ( w, callData );  
}
```

```
void Bb3DUI::doOptionFlowReverseCallback ( Widget      w,  
                                          XtPointer clientData,  
                                          XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionFlowReverse ( w, callData );  
}
```

```
void Bb3DUI::doOptionOrthoCameraCallback ( Widget      w,  
                                          XtPointer clientData,  
                                          XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionOrthoCamera ( w, callData );  
}
```

```
void Bb3DUI::doOptionPersCameraCallback ( Widget      w,  
                                          XtPointer clientData,  
                                          XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionPersCamera ( w, callData );  
}
```

```
void Bb3DUI::doOptionPersCameraRotCallback ( Widget      w,  
                                             XtPointer clientData,  
                                             XtPointer callData )
```

```
{  
    Bb3DUI* obj = ( Bb3DUI * ) clientData;  
    obj->doOptionPersCameraRot ( w, callData );  
}
```

```

}

void Bb3DUI::doOptionSetting3DCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    Bb3DUI* obj = ( Bb3DUI * ) clientData;
    obj->doOptionSetting3D ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void Bb3DUI::HeightDn ( Widget, XtPointer )
{
    // This virtual function is called from HeightDnCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::HeightUp ( Widget, XtPointer )
{
    // This virtual function is called from HeightUpCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::High3D ( Widget, XtPointer )
{
    // This virtual function is called from High3DCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::Low3D ( Widget, XtPointer )
{
    // This virtual function is called from Low3DCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::TextHeight3D ( Widget, XtPointer )
{
    // This virtual function is called from TextHeight3DCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::TextYPos ( Widget, XtPointer )
{
    // This virtual function is called from TextYPosCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::YPosDn ( Widget, XtPointer )
{
    // This virtual function is called from YPosDnCallback.
    // This function is normally overridden by a derived class.
}

```

```

void Bb3DUI::YPosUp ( Widget, XtPointer )
{
    // This virtual function is called from YPosUpCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doButtonNormalize ( Widget, XtPointer )
{
    // This virtual function is called from doButtonNormalizeCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionFixed ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFixedCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionFlowASIS ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFlowASISCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionFlowReverse ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFlowReverseCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionOrthoCamera ( Widget, XtPointer )
{
    // This virtual function is called from doOptionOrthoCameraCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionPersCamera ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPersCameraCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionPersCameraRot ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPersCameraRotCallback.
    // This function is normally overridden by a derived class.
}

void Bb3DUI::doOptionSetting3D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionSetting3DCallback.
    // This function is normally overridden by a derived class.
}

```

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbAnimation
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbAnimationUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbAnimation.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbAnimationUI and are
// available as protected data members inherited by this class
//
// XmToggleButton          _toggleFlow
// XmToggleButton          _toggle2D
// XmToggleButton          _toggle3D
// XmToggleButton          _toggleSymphony
// XmPushButton            _buttonHeart
// XmLabel                  _labelTime
// XmTextField              _textfield
//
////////////////////////////////////

///----- Start editable code block: headers and declarations

#include "Animate.h"
#include "Utility.h"
#include <Vk/VkFormat.h>

///----- End editable code block: headers and declarations

///----- BbAnimation Constructor

BbAnimation::BbAnimation(const char *name, Widget parent) :
    BbAnimationUI(name, parent)

```

```

{
    // This constructor calls BbAnimationUI(parent, name)
    // which calls BbAnimationUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbAnimation constructor

    //---- End editable code block: BbAnimation constructor

}    // End Constructor

BbAnimation::BbAnimation(const char *name) :
    BbAnimationUI(name)
{
    // This constructor calls BbAnimationUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbAnimation constructor 2

    //---- End editable code block: BbAnimation constructor 2

}    // End Constructor

BbAnimation::~BbAnimation()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbAnimation destructor

    //---- End editable code block: BbAnimation destructor

}    // End Destructor

const char * BbAnimation::className() // classname
{
    return ("BbAnimation");
} // End className()

void BbAnimation::animateTime ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation animateTime

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbAnimation::animateTime is implemented:

```

```

//::VkUnimplemented ( w, "BbAnimation::animateTime" );

if(_animate != NULL)
    _animate->_msec = atof(XmTextFieldGetString(get_textfield()));

//---- End editable code block: BbAnimation animateTime
} // End BbAnimation::animateTime()

void BbAnimation::doButtonHeart ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation doButtonHeart
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbAnimation::doButtonHeart is implemented
    //::VkUnimplemented ( w, "BbAnimation::doButtonHeart" );

    float bpm = _objMag -> msgsRight.HR;
    int msec = (int)(60.0/bpm/(float)_animate->_num_imgs * 1000.0);

    XmTextFieldSetString(get_textfield(), (char *)VkFormat("%d", msec ));

    if(_animate != NULL)
        _animate -> _msec = msec;

    //---- End editable code block: BbAnimation doButtonHeart
} // End BbAnimation::doButtonHeart()

void BbAnimation::setToggle1D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation setToggle1D
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbAnimation::setToggle1D is implemented:
    //::VkUnimplemented ( w, "BbAnimation::setToggle1D" );
    _objMag -> msgsRight.animate_mode = ANIMATE_1D;
    //---- End editable code block: BbAnimation setToggle1D
} // End BbAnimation::setToggle1D()

void BbAnimation::setToggle2D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation setToggle2D
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbAnimation::setToggle2D is implemented:
    //::VkUnimplemented ( w, "BbAnimation::setToggle2D" );

    if(_objMag -> msgsRight.img_space == IMAGE_2D)
        _objMag -> msgsRight.animate_mode = ANIMATE_2D;

    //---- End editable code block: BbAnimation setToggle2D
}

```

```
} // End BbAnimation::Toggle2D()
```

```
void BbAnimation::setToggle3D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation setToggle3D

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbAnimation::setToggle3D is implemented:
    //::VkUnimplemented ( w, "BbAnimation::setToggle3D" );

    if(_objMag -> msgsRight.img_space == IMAGE_3D)
        _objMag -> msgsRight.animate_mode = ANIMATE_3D;

    //---- End editable code block: BbAnimation setToggle3D
} // End BbAnimation::setToggle3D()
```

```
void BbAnimation::setToggleSymphony ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbAnimation setToggleSymphony

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbAnimation::setToggleSymphony is impleme
    //::VkUnimplemented ( w, "BbAnimation::setToggleSymphony" );

    if(_objMag -> msgsRight.img_space == IMAGE_2D)
        _objMag -> msgsRight.animate_mode = ANIMATE_SYMPHONY;

    //---- End editable code block: BbAnimation setToggleSymphony
} // End BbAnimation::setToggleSymphony()
```

```
////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////
```

```
VkComponent *BbAnimation::CreateBbAnimation( const char *name, Widget parent )
{
    VkComponent *obj = new BbAnimation ( name, parent );
    return ( obj );
} // End CreateBbAnimation
```

```
////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////
```

```
// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details
```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *BbAnimation::RegisterBbAnimationInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int                (Use XmRInt)
    //     float             (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg")
    //     A filename        (Use VkRFilename or "Filename")
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //----- Start editable code block: BbAnimationUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //----- End editable code block: BbAnimationUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbAnimationInterface()

//----- End of generated code

//----- Start editable code block: End of generated code

void BbAnimation::init()
{
    XmToggleButtonSetState(_toggle2D, TRUE, TRUE);
}

void BbAnimation::set_toggle(int which)
{
    if(which == 2)
        XmToggleButtonSetState(_toggle2D, TRUE, TRUE);
    if(which == 3)
        XmToggleButtonSetState(_toggle3D, TRUE, TRUE);
}

```

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbAnimationUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
-//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbAnimationUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
/*--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbAnimationUI::_defaultBbAnimationUIResources[] = {
    "*buttonHeart.labelString: Heart Rate",
    "*labelTime.labelString: Time",
    "*tabLabel: Animation",
    "*toggle2D.labelString: 2D",
    "*toggle3D.labelString: 3D Pulsatile",
    "*toggleFlow.labelString: 1D Flow Waveform",
    "*toggleSymphony.labelString: Symphony",

    /*--- Start editable code block: BbAnimationUI Default Resources

    /*--- End editable code block: BbAnimationUI Default Resources

    (char*)NULL
};

BbAnimationUI::BbAnimationUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.

```


[illegible]

```
installDestroyHandler();
```

```
// Create widgets used in this component  
// All variables are data members of this class
```

```
_radioboxAnimate = XtVaCreateManagedWidget ( "radioboxAnimate",  
                                              xmRowColumnWidgetClass,  
                                              _baseWidget,  
                                              XmNpacking, XmPACK_COLUMN,  
                                              XmNradioBehavior, True,  
                                              XmNradioAlwaysOne, True,  
                                              XmNx, 348,  
                                              XmNy, 12,  
                                              XmNwidth, 152,  
                                              XmNheight, 119,  
                                              (XtPointer) NULL );
```

```
_toggleFlow = XtVaCreateManagedWidget ( "toggleFlow",  
                                         xmToggleButtonWidgetClass,  
                                         _radioboxAnimate,  
                                         XmNlabelType, XmSTRING,  
                                         (XtPointer) NULL );
```

```
XtAddCallback ( _toggleFlow,  
               XmNvalueChangedCallback,  
               &BbAnimationUI::setToggle1DCallback,  
               (XtPointer) this );
```

```
_toggle2D = XtVaCreateManagedWidget ( "toggle2D",  
                                       xmToggleButtonWidgetClass,  
                                       _radioboxAnimate,  
                                       XmNlabelType, XmSTRING,  
                                       (XtPointer) NULL );
```

```
XtAddCallback ( _toggle2D,  
               XmNvalueChangedCallback,  
               &BbAnimationUI::setToggle2DCallback,  
               (XtPointer) this );
```

```
_toggle3D = XtVaCreateManagedWidget ( "toggle3D",  
                                       xmToggleButtonWidgetClass,  
                                       _radioboxAnimate,  
                                       XmNlabelType, XmSTRING,  
                                       (XtPointer) NULL );
```

```
XtAddCallback ( _toggle3D,  
               XmNvalueChangedCallback,  
               &BbAnimationUI::setToggle3DCallback,  
               (XtPointer) this );
```

```
_toggleSymphony = XtVaCreateManagedWidget ( "toggleSymphony",  
                                              xmToggleButtonWidgetClass,  
                                              _radioboxAnimate,  
                                              XmNlabelType, XmSTRING,  
                                              (XtPointer) NULL );
```

```
XtAddCallback ( _toggleSymphony,  
               XmNvalueChangedCallback,
```

```
_buttonHeart = XtVaCreateManagedWidget ( "buttonHeart",
                                           xmPushButtonWidgetClass,
                                           _baseWidget,
                                           XmNlabelType, XmSTRING,
                                           XmNx, 162,
                                           XmNy, 82,
                                           XmNwidth, 90,
                                           XmNheight, 40,
                                           (XtPointer) NULL );
```

```
XtAddCallback ( _buttonHeart,
                 XmNactivateCallback,
                 &BbAnimationUI::doButtonHeartCallback,
                 (XtPointer) this );
```

```
_labelTime = XtVaCreateManagedWidget ( "labelTime",
                                          xmLabelWidgetClass,
                                          _baseWidget,
                                          XmNlabelType, XmSTRING,
                                          XmNx, 186,
                                          XmNy, 10,
                                          XmNwidth, 36,
                                          XmNheight, 20,
                                          (XtPointer) NULL );
```

```
_textfield = XtVaCreateManagedWidget ( "textfield",
                                          xmTextFieldWidgetClass,
                                          _baseWidget,
                                          XmNcolumns, 7,
                                          XmNx, 168,
                                          XmNy, 36,
                                          XmNheight, 40,
                                          (XtPointer) NULL );
```

```
XtAddCallback ( _textfield,
                 XmNactivateCallback,
                 &BbAnimationUI::animateTimeCallback,
                 (XtPointer) this );
```

```
//---- Start editable code block: BbAnimationUI create
```

```
//---- End editable code block: BbAnimationUI create
```

```
}
```

```
const char * BbAnimationUI::className()
{
    return ("BbAnimationUI");
} // End className()
```

```
////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////
```

```

void BbAnimationUI::animateTimeCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->animateTime ( w, callData );
}

void BbAnimationUI::doButtonHeartCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->doButtonHeart ( w, callData );
}

void BbAnimationUI::setToggle1DCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->setToggle1D ( w, callData );
}

void BbAnimationUI::setToggle2DCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->setToggle2D ( w, callData );
}

void BbAnimationUI::setToggle3DCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->setToggle3D ( w, callData );
}

void BbAnimationUI::setToggleSymphonyCallback ( Widget    w,
                                                XtPointer clientData,
                                                XtPointer callData )

{
    BbAnimationUI* obj = ( BbAnimationUI * ) clientData;
    obj->setToggleSymphony ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbAnimationUI::animateTime ( Widget, XtPointer )
{
    // This virtual function is called from animateTimeCallback.
    // This function is normally overridden by a derived class.
}

void BbAnimationUI::doButtonHeart ( Widget, XtPointer )
{
    // This virtual function is called from doButtonHeartCallback.
    // This function is normally overridden by a derived class.
}

```

```
}

void BbAnimationUI::setToggle1D ( Widget, XtPointer )
{
    // This virtual function is called from setToggle1DCallback.
    // This function is normally overridden by a derived class.
}

void BbAnimationUI::setToggle2D ( Widget, XtPointer )
{
    // This virtual function is called from setToggle2DCallback.
    // This function is normally overridden by a derived class.
}

void BbAnimationUI::setToggle3D ( Widget, XtPointer )
{
    // This virtual function is called from setToggle3DCallback.
    // This function is normally overridden by a derived class.
}

void BbAnimationUI::setToggleSymphony ( Widget, XtPointer )
{
    // This virtual function is called from setToggleSymphonyCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbDetail
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbDetailUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "BbDetail.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>

```

```

extern void VkUnimplemented ( Widget, const char * );

```

```

////////////////////////////////////
// The following non-container elements are created by BbDetailUI and are
// available as protected data members inherited by this class
//
// XmToggleButton          _toggleShow
// XmToggleButton          _toggleHide
// XmTextField              _textfieldZ
// XmLabel                  _labelZ
// XmTextField              _textfieldSignal
// XmTextField              _textfieldB
// XmTextField              _textfieldG
// XmTextField              _textfieldR
// XmTextField              _textfieldY
// XmLabel                  _labelSignal
// XmLabel                  _labelB
// XmLabel                  _labelG
// XmLabel                  _labelR
// XmLabel                  _labelY
// XmLabel                  _labelX
// XmTextField              _textfieldX
//
////////////////////////////////////

```

```

//----- Start editable code block: headers and declarations

```

```

#include <Vk/VkFormat.h>

```

```

//----- End editable code block: headers and declarations

```

```
//---- BbDetail Constructor
```

```
BbDetail::BbDetail(const char *name, Widget parent) :
    BbDetailUI(name, parent)
{
    // This constructor calls BbDetailUI(parent, name)
    // which calls BbDetailUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbDetail constructor

    //---- End editable code block: BbDetail constructor

}    // End Constructor
```

```
BbDetail::BbDetail(const char *name) :
    BbDetailUI(name)
{
    // This constructor calls BbDetailUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbDetail constructor 2

    //---- End editable code block: BbDetail constructor 2

}    // End Constructor
```

```
BbDetail::~BbDetail()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbDetail destructor

    //---- End editable code block: BbDetail destructor

}    // End Destructor
```

```
const char * BbDetail::className() // classname
{
    return ("BbDetail");
} // End className()
```

```
void BbDetail::setToggleHide ( Widget w, XtPointer callData )
{
```

```

//---- Start editable code block: BbDetail setToggleHide
XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

//--- Comment out the following line when BbDetail::setToggleHide is implemented:
//::VkUnimplemented ( w, "BbDetail::setToggleHide" );
_objMag -> msgsRight.show_detail = FALSE;

//---- End editable code block: BbDetail setToggleHide
} // End BbDetail::setToggleHide()

void BbDetail::setToggleShow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDetail setToggleShow

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDetail::setToggleShow is implemented:
    //::VkUnimplemented ( w, "BbDetail::setToggleShow" );

    _objMag -> msgsRight.show_detail = TRUE;

    //---- End editable code block: BbDetail setToggleShow
} // End BbDetail::setToggleShow()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *BbDetail::CreateBbDetail( const char *name, Widget parent )
{
    VkComponent *obj = new BbDetail ( name, parent );
    return ( obj );
} // End CreateBbDetail

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```



```

void *BbDetail::RegisterBbDetailInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char *      (Use XmRString)
    // Boolean           (Use XmRBoolean)
    // int               (Use XmRInt)
    // float             (Use XmRFloat)
    // No argument       (Use VkRNoArg or "NoArg")
    // A filename        (Use VkRFilename or "Filename")
    // An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    // A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbDetailUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbDetailUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbDetailInterface()


//---- End of generated code

//---- Start editable code block: End of generated code

void BbDetail::init()
{
    XmToggleButtonSetState(_toggleHide, TRUE, TRUE);
}

void BbDetail::set(int x, int y, int r, int g, int b, int signal)
{
    XmTextFieldSetString(_textfieldX, (char *)VkFormat("%d", x));
    XmTextFieldSetString(_textfieldY, (char *)VkFormat("%d", y));
    XmTextFieldSetString(_textfieldR, (char *)VkFormat("%d", r));
    XmTextFieldSetString(_textfieldG, (char *)VkFormat("%d", g));
    XmTextFieldSetString(_textfieldB, (char *)VkFormat("%d", b));
    XmTextFieldSetString(_textfieldSignal, (char *)VkFormat("%d", signal));
}

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbDetailUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbDetailUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
///--- Start editable code block: headers and declarations

///--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbDetailUI::_defaultBbDetailUIResources[] = {
    "*labelB.labelString: B",
    "*labelG.labelString: G",
    "*labelR.labelString: R",
    "*labelSignal.labelString: Signal",
    "*labelX.labelString: X",
    "*labelY.labelString: Y",
    "*labelZ.labelString: Z",
    "*tabLabel: Pixel",
    "*toggleHide.labelString: Hide",
    "*toggleShow.labelString: Show",

    ///--- Start editable code block: BbDetailUI Default Resources

    ///--- End editable code block: BbDetailUI Default Resources

    (char*)NULL
};

BbDetailUI::BbDetailUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.

```



```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component  
// All variables are data members of this class
```

```
_radioboxPixel = XtVaCreateManagedWidget ( "radioboxPixel",  
                                             xmRowColumnWidgetClass,  
                                             _baseWidget,  
                                             XmNorientation, XmHORIZONTAL,  
                                             XmNpacking, XmPACK_COLUMN,  
                                             XmNradioBehavior, True,  
                                             XmNradioAlwaysOne, True,  
                                             XmNx, 407,  
                                             XmNy, 78,  
                                             XmNwidth, 133,  
                                             XmNheight, 32,  
                                             (XtPointer) NULL );
```

```
_toggleShow = XtVaCreateManagedWidget ( "toggleShow",  
                                          xmToggleButtonWidgetClass,  
                                          _radioboxPixel,  
                                          XmNlabelType, XmSTRING,  
                                          (XtPointer) NULL );
```

```
XtAddCallback ( _toggleShow,  
               XmNvalueChangedCallback,  
               &BbDetailUI::setToggleShowCallback,  
               (XtPointer) this );
```

```
_toggleHide = XtVaCreateManagedWidget ( "toggleHide",  
                                          xmToggleButtonWidgetClass,  
                                          _radioboxPixel,  
                                          XmNlabelType, XmSTRING,  
                                          (XtPointer) NULL );
```

```
XtAddCallback ( _toggleHide,  
               XmNvalueChangedCallback,  
               &BbDetailUI::setToggleHideCallback,  
               (XtPointer) this );
```

```
_textfieldZ = XtVaCreateManagedWidget ( "textfieldZ",  
                                          xmTextFieldWidgetClass,  
                                          _baseWidget,  
                                          XmNcolumns, 7,  
                                          XmNx, 102,  
                                          XmNy, 91,  
                                          XmNheight, 35,  
                                          (XtPointer) NULL );
```

```
_labelZ = XtVaCreateManagedWidget ( "labelZ",  
                                     xmLabelWidgetClass,  
                                     _baseWidget,  
                                     XmNlabelType, XmSTRING,  
                                     XmNx, 62,  
                                     XmNy, 97,  
                                     XmNwidth, 13,
```

```
_textfieldSignal = XtVaCreateManagedWidget ( "textfieldSignal",  
                                              xmTextFieldWidgetClass,  
                                              _baseWidget,  
                                              XmNcolumns, 7,  
                                              XmNx, 465,  
                                              XmNy, 24,  
                                              XmNheight, 35,  
                                              (XtPointer) NULL );
```

```
_textfieldB = XtVaCreateManagedWidget ( "textfieldB",  
                                          xmTextFieldWidgetClass,  
                                          _baseWidget,  
                                          XmNcolumns, 7,  
                                          XmNx, 275,  
                                          XmNy, 90,  
                                          XmNheight, 35,  
                                          (XtPointer) NULL );
```

```
_textfieldG = XtVaCreateManagedWidget ( "textfieldG",  
                                          xmTextFieldWidgetClass,  
                                          _baseWidget,  
                                          XmNcolumns, 7,  
                                          XmNx, 275,  
                                          XmNy, 50,  
                                          XmNheight, 35,  
                                          (XtPointer) NULL );
```

```
_textfieldR = XtVaCreateManagedWidget ( "textfieldR",  
                                          xmTextFieldWidgetClass,  
                                          _baseWidget,  
                                          XmNcolumns, 7,  
                                          XmNx, 274,  
                                          XmNy, 12,  
                                          XmNheight, 35,  
                                          (XtPointer) NULL );
```

```
_textfieldY = XtVaCreateManagedWidget ( "textfieldY",  
                                          xmTextFieldWidgetClass,  
                                          _baseWidget,  
                                          XmNcolumns, 7,  
                                          XmNx, 101,  
                                          XmNy, 51,  
                                          XmNheight, 35,  
                                          (XtPointer) NULL );
```

```
_labelSignal = XtVaCreateManagedWidget ( "labelSignal",  
                                           xmLabelWidgetClass,  
                                           _baseWidget,  
                                           XmNlabelType, XmSTRING,  
                                           XmNx, 402,  
                                           XmNy, 33,  
                                           XmNwidth, 48,  
                                           XmNheight, 20,  
                                           (XtPointer) NULL );
```

```
_labelB = XtVaCreateManagedWidget ( "labelB",
```

```

xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 236,
XmNy, 96,
XmNwidth, 14,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelG = XtVaCreateManagedWidget ( "labelG",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 234,
XmNy, 56,
XmNwidth, 15,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelR = XtVaCreateManagedWidget ( "labelR",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 234,
XmNy, 20,
XmNwidth, 15,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelY = XtVaCreateManagedWidget ( "labelY",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 63,
XmNy, 58,
XmNwidth, 14,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelX = XtVaCreateManagedWidget ( "labelX",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 64,
XmNy, 18,
XmNwidth, 13,
XmNheight, 20,
(XtPointer) NULL );

```

```

_textfieldX = XtVaCreateManagedWidget ( "textfieldX",
xmTextFieldWidgetClass,
_baseWidget,
XmNcolumns, 7,
XmNx, 100,
XmNy, 11,
XmNheight, 35,
(XtPointer) NULL );

```

//---- Start editable code block: BbDetailUI create

```

//----- End editable code block: BbDetailUI create
}

const char * BbDetailUI::className()
{
    return ("BbDetailUI");
} // End className()

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

void BbDetailUI::setToggleHideCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbDetailUI* obj = ( BbDetailUI * ) clientData;
    obj->setToggleHide ( w, callData );
}

void BbDetailUI::setToggleShowCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbDetailUI* obj = ( BbDetailUI * ) clientData;
    obj->setToggleShow ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbDetailUI::setToggleHide ( Widget, XtPointer )
{
    // This virtual function is called from setToggleHideCallback.
    // This function is normally overridden by a derived class.
}

void BbDetailUI::setToggleShow ( Widget, XtPointer )
{
    // This virtual function is called from setToggleShowCallback.
    // This function is normally overridden by a derived class.
}

//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

```

User: meide
Host: phoenix
Class: phoenix
Job: main.C

251


```

////////////////////////////////////
//
// Source file for BbDisplay
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbDisplayUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbDisplay.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/RowColumn.h>
#include <Xm/Separator.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbDisplayUI and are
// available as protected data members inherited by this class
//
// XmToggleButton          _toggleCombo
// XmToggleButton          _toggleNormal
// VkOptionMenu *           _optionMenu3
// VkMenuItem *             _optionWhole2D
// VkMenuItem *             _optionL3D
// VkMenuItem *             _optionFlow2D
// VkMenuItem *             _optionFlow3D
// XmToggleButton          _toggleLeft
// XmToggleButton          _toggleRight
// XmToggleButton          _toggleBoth
// VkOptionMenu *           _optionMenuZoom
// VkMenuItem *             _optionNormal
// VkMenuItem *             _optionDouble
// VkMenuItem *             _optionHalf
// XmLabel                  _labelZoom
// XmSeparator              _separator2
// XmLabel                  _labelImageNumber
// XmTextField              _textfieldZoom
// XmLabel                  _labelDisplayTotalNum
// XmLabel                  _labelDisplayTotal
// XmTextField              _textfieldDisplayImgNumber
//
////////////////////////////////////

```

```

//---- Start editable code block: headers and declarations

#include <stdlib.h>
#include "Utility_Widget.h"
#include "Utility.h"

//---- End editable code block: headers and declarations

//---- BbDisplay Constructor

BbDisplay::BbDisplay(const char *name, Widget parent) :
    BbDisplayUI(name, parent)
{
    // This constructor calls BbDisplayUI(parent, name)
    // which calls BbDisplayUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbDisplay constructor

    //---- End editable code block: BbDisplay constructor

}    // End Constructor

BbDisplay::BbDisplay(const char *name) :
    BbDisplayUI(name)
{
    // This constructor calls BbDisplayUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbDisplay constructor 2

    //---- End editable code block: BbDisplay constructor 2

}    // End Constructor

BbDisplay::~BbDisplay()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbDisplay destructor

    //---- End editable code block: BbDisplay destructor

}    // End Destructor

```

```
const char * BbDisplay::className() // classname
{
    return ("BbDisplay");
} // End className()
```

```
void BbDisplay::doOptionDouble ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbDisplay doOptionDouble

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionDouble is implemented:

    //::VkUnimplemented ( w, "BbDisplay::doOptionDouble" );
    Utility_Widget *u = new Utility_Widget();

    if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
    {
        _objMag -> msgsLeft.img_zoom *= 2.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_RIGHT)
    {
        _objMag -> msgsRight.img_zoom *= 2.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsRight.img_zoom);
        _objMag -> update_RimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_BOTH)
    {
        _objMag -> msgsLeft.img_zoom *= 2.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
        _objMag -> msgsRight.img_zoom *= 2.0;
        _objMag -> update_RimgView();
    }

    delete u;

    //----- End editable code block: BbDisplay doOptionDouble
} // End BbDisplay::doOptionDouble()
```

```
void BbDisplay::doOptionFlow2D ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbDisplay doOptionFlow2D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionFlow2D is implemented:

    //::VkUnimplemented ( w, "BbDisplay::doOptionFlow2D" );

    /*
    if(_objMag->msgsRight.img_select == RIGHT_IMG_ROI)
    {
        _objMag->msgsLeft.img_space = IMAGE_FLOW2D;
        _objMag -> update_LimgView();
    }
    */
```

```

//---- End editable code block: BbDisplay doOptionFlow2D
} // End BbDisplay::doOptionFlow2D()

void BbDisplay::doOptionFlow3D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay doOptionFlow3D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionFlow3D is implemented:
    //::VkUnimplemented ( w, "BbDisplay::doOptionFlow3D" );

    /*
    if(_objMag->msgsRight.img_select == RIGHT_IMG_ROI)
    {
        _objMag->msgsLeft.img_space = IMAGE_FLOW3D;
        _objMag -> update_LimgView();
    }
    */

    //---- End editable code block: BbDisplay doOptionFlow3D
} // End BbDisplay::doOptionFlow3D()

void BbDisplay::doOptionHalf ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay doOptionHalf

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionHalf is implemented:
    //::VkUnimplemented ( w, "BbDisplay::doOptionHalf" );
    Utility_Widget *u = new Utility_Widget();

    if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
    {
        _objMag -> msgsLeft.img_zoom *= 0.5;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_RIGHT)
    {
        _objMag -> msgsRight.img_zoom *= 0.5;
        u->set_textfield(_textfieldZoom, _objMag -> msgsRight.img_zoom);
        _objMag -> update_RimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_BOTH)
    {
        _objMag -> msgsLeft.img_zoom *= 0.5;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
        _objMag -> msgsRight.img_zoom *= 0.5;
        _objMag -> update_RimgView();
    }

    delete u;

    //---- End editable code block: BbDisplay doOptionHalf

```

```
} // End BbDisplay::doOptionHalf()
```

```
void BbDisplay::doOptionL3D ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbDisplay doOptionL3D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionL3D is implemented:
    //::VkUnimplemented ( w, "BbDisplay::doOptionL3D" );

    if(_objMag->msgsRight.img_select == RIGHT_IMG_ROI)
    {
        _objMag->msgsLeft.img_space = IMAGE_3D;
        _objMag -> update_LimgView();
    }

    //----- End editable code block: BbDisplay doOptionL3D
} // End BbDisplay::doOptionL3D()

void BbDisplay::doOptionNormal ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbDisplay doOptionNormal

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionNormal is implemented:
    //::VkUnimplemented ( w, "BbDisplay::doOptionNormal" );
    Utility_Widget *u = new Utility_Widget();

    if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
    {
        _objMag -> msgsLeft.img_zoom = 1.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_RIGHT)
    {
        _objMag -> msgsRight.img_zoom = 1.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsRight.img_zoom);
        _objMag -> update_RimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_BOTH)
    {
        _objMag -> msgsLeft.img_zoom = 1.0;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
        _objMag -> msgsRight.img_zoom = 1.0;
        _objMag -> update_RimgView();
    }

    delete u;

    //----- End editable code block: BbDisplay doOptionNormal
} // End BbDisplay::doOptionNormal()
```

```

void BbDisplay::doOptionWhole2D ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay doOptionWhole2D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::doOptionWhole2D is implemented
    //::VkUnimplemented ( w, "BbDisplay::doOptionWhole2D" );

    _objMag->msgsLeft.img_space = IMAGE_2D;

    _objMag -> update_LimgView();

    //---- End editable code block: BbDisplay doOptionWhole2D
}    // End BbDisplay::doOptionWhole2D()

```

```

void BbDisplay::imgNum ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay imgNum

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::imgNum is implemented:
    //::VkUnimplemented ( w, "BbDisplay::imgNum" );

    int img_number = atoi(XmTextFieldGetString(_textfieldDisplayImgNumber));

    if(img_number < _objMag->msgsLoaded.img_start)
        img_number = _objMag->msgsLoaded.img_end;
    if(img_number > _objMag-> msgsLoaded.img_end)
        img_number = _objMag->msgsLoaded.img_start;

    _objMag->update_Aimg(img_number);

    //---- End editable code block: BbDisplay imgNum
}    // End BbDisplay::imgNum()

```

```

void BbDisplay::imgZoom ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay imgZoom

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::imgZoom is implemented:
    //::VkUnimplemented ( w, "BbDisplay::imgZoom" );

    float zoom = atof(XmTextFieldGetString(_textfieldZoom));

    Utility_Widget *u = new Utility_Widget();

    if(_objMag -> msgsLeft.img_zoom_select == ZOOM_LEFT)
    {
        _objMag -> msgsLeft.img_zoom = zoom;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_RIGHT)
    {
        _objMag -> msgsRight.img_zoom = zoom;
    }
}

```

```

        u->set_textfield(_textfieldZoom, _objMag -> msgsRight.img_zoom);
        _objMag -> update_LimgView();
    }
    else if(_objMag -> msgsLeft.img_zoom_select == ZOOM_BOTH)
    {
        _objMag -> msgsLeft.img_zoom = zoom;
        u->set_textfield(_textfieldZoom, _objMag -> msgsLeft.img_zoom);
        _objMag -> update_LimgView();
        _objMag -> msgsRight.img_zoom = zoom;
        _objMag -> update_RimgView();
    }

    delete u;

    //---- End editable code block: BbDisplay imgZoom
} // End BbDisplay::imgZoom()

void BbDisplay::setToggleBoth ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay setToggleBoth
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::setToggleBoth is implemented:

    //::VkUnimplemented ( w, "BbDisplay::setToggleBoth" );
    _objMag -> msgsLeft.img_zoom_select = ZOOM_BOTH;

    //---- End editable code block: BbDisplay setToggleBoth
} // End BbDisplay::setToggleBoth()

void BbDisplay::setToggleCombo ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay setToggleCombo
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::setToggleCombo is implemented:

    //::VkUnimplemented ( w, "BbDisplay::setToggleCombo" );
    _objMag -> msgsLeft.layout = LAYOUT_COMBO;

    //---- End editable code block: BbDisplay setToggleCombo
} // End BbDisplay::setToggleCombo()

void BbDisplay::setToggleLeft ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay setToggleLeft
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbDisplay::setToggleLeft is implemented:

    //::VkUnimplemented ( w, "BbDisplay::setToggleLeft" );
    _objMag -> msgsLeft.img_zoom_select = ZOOM_LEFT;
    Utility_Widget *u = new Utility_Widget();

```

```
u->set_textfield(_textfieldZoom,_objMag -> msgsLeft.img_zoom);
delete u;
```

259

```
//---- End editable code block: BbDisplay setToggleLeft
} // End BbDisplay::setToggleLeft()

void BbDisplay::setToggleNormal ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay setToggleNormal
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbDisplay::setToggleNormal is implemented
    //::VkUnimplemented ( w, "BbDisplay::setToggleNormal" );
    _objMag -> msgsLeft.layout = LAYOUT_NORMAL;

    //---- End editable code block: BbDisplay setToggleNormal
} // End BbDisplay::setToggleNormal()

void BbDisplay::setToggleRight ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbDisplay setToggleRight
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbDisplay::setToggleRight is implemented:
    //::VkUnimplemented ( w, "BbDisplay::setToggleRight" );
    _objMag -> msgsLeft.img_zoom_select = ZOOM_RIGHT;
    Utility_Widget *u = new Utility_Widget();
    u->set_textfield(_textfieldZoom,_objMag -> msgsRight.img_zoom);
    delete u;

    //---- End editable code block: BbDisplay setToggleRight
} // End BbDisplay::setToggleRight()
```

```
////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////
```

```
VkComponent *BbDisplay::CreateBbDisplay( const char *name, Widget parent )
{
    VkComponent *obj = new BbDisplay ( name, parent );
    return ( obj );
} // End CreateBbDisplay
```

```
////////////////////////////////////
// Function for accessing a description of the dynamic interface
```



```
// to this class.
```

260

```
////////////////////////////////////  
// WARNING: This structure is different than that used with 1.1 RapidApp.  
// See the RapidApp release notes for details
```

```
struct InterfaceMap {  
    char *resourceName;  
    char *methodName;  
    char *argType;  
    char *definingClass; // Optional, if not this class  
    void (VkCallbackObject::*method)(...); // Reserved, do not set  
};
```

```
void *BbDisplay::RegisterBbDisplayInterface()  
{  
    // This structure registers information about this class  
    // that allows RapidApp to create and manipulate an instance.  
    // Each entry provides a resource name that will appear in the  
    // resource manager palette when an instance of this class is  
    // selected, the name of the member function as a string,  
    // the type of the single argument to this function, and an.  
    // optional argument indicating the class that defines this function.  
    // All member functions must have the form  
    //  
    //     void memberFunction ( Type );  
    //  
    // where "Type" is one of:  
    //     const char *      (Use XmRString)  
    //     Boolean           (Use XmRBoolean)  
    //     int               (Use XmRInt)  
    //     float             (Use XmRFloat)  
    //     No argument       (Use VkRNoArg or "NoArg")  
    //     A filename        (Use VkRFilename or "Filename")  
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")  
    //     A callback        (Use XmRCallback)  
  
    static InterfaceMap map[] = {  
        //---- Start editable code block: BbDisplayUI resource table  
  
        // { "resourceName", "setAttribute", XmRString},  
        //---- End editable code block: BbDisplayUI resource table  
        { NULL }, // MUST be NULL terminated  
    };  
  
    return map;  
} // End RegisterBbDisplayInterface()
```

```
//---- End of generated code
```

```
//---- Start editable code block: End of generated code
```

```
void BbDisplay::init()  
{  
    XmToggleButtonSetState(_toggleBoth, TRUE, TRUE);  
}
```

```
//---- End editable code block: End of generated code
```



```

////////////////////////////////////
//
// Source file for BbDisplayUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

```

```
#include "BbDisplayUI.h" // Generated header file for this class
```

```

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/RowColumn.h>
#include <Xm/Separator.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
/*--- Start editable code block: headers and declarations

```

```
/*--- End editable code block: headers and declarations
```

```

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

```

```

String BbDisplayUI::_defaultBbDisplayUIResources[] = {
    "*labelDisplayTotal.labelString: Total",
    "*labelDisplayTotalNum.labelString: 1",
    "*labelImageNumber.labelString: Image",
    "*labelZoom.labelString: Zoom",
    "*optionDouble.labelString: Double",
    "*optionFlow2D.labelString: Flow 2D",
    "*optionFlow3D.labelString: Flow 3D",
    "*optionHalf.labelString: Half",
    "*optionL3D.labelString: ROI 3D",
    "*optionMenuZoom.labelString: ",
    "*optionNormal.labelString: Normal",
    "*optionWhole2D.labelString: ASIS 2D",
    "**tabLabel: Display",
    "**textfieldDisplayImgNumber.value: 1",
    "**textfieldZoom.value: 1",
    "**toggleBoth.labelString: Both",
    "**toggleCombo.labelString: Combo",
    "**toggleLeft.labelString: Left",
    "**toggleNormal.labelString: Normal",

```

```
    **toggleRight.labelString: Right",
    "+*labelDisplayText.fontList: SGI_DYNAMIC SmallInLabelFont", 263
```

```
//----- Start editable code block: BbDisplayUI Default Resources
```

```
//----- End editable code block: BbDisplayUI Default Resources
```

```
(char*)NULL
```

```
};
```

```
BbDisplayUI::BbDisplayUI ( const char *name ) : VkComponent ( name )
```

```
{
```

```
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.
```

```
//----- Start editable code block: BbDisplay constructor 2
```

```
//----- End editable code block: BbDisplay constructor 2
```

```
}    // End Constructor
```

```
BbDisplayUI::BbDisplayUI ( const char *name, Widget parent ) : VkComponent ( name )
```

```
{
```

```
    //----- Start editable code block: BbDisplay pre-create
```

```
//----- End editable code block: BbDisplay pre-create
```

```
    // Call creation function to build the widget tree.
```

```
    create ( parent );
```

```
//----- Start editable code block: BbDisplay constructor
```

```
//----- End editable code block: BbDisplay constructor
```

```
}    // End Constructor
```

```
BbDisplayUI::~BbDisplayUI()
```

```
{
```

```
    // Base class destroys widgets
```

```
//----- Start editable code block: BbDisplayUI destructor
```

```
//----- End editable code block: BbDisplayUI destructor
```

```
}    // End destructor
```

```
void BbDisplayUI::create ( Widget parent )
```

```
{
```

```
Arg      args[10];
Cardinal count;
count = 0;
```

264

```
// Load any class-defaulted resources for this object
```

```
setDefaultResources ( parent, _defaultBbDisplayUIResources );
```

```
// Create an unmanaged widget as the top of the widget hierarchy
```

```
_baseWidget = _bbDisplay = XtVaCreateWidget ( _name,
                                              xmBulletinBoardWidgetClass,
                                              parent,
                                              XmNresizePolicy, XmRESIZE_GROW,
                                              (XtPointer) NULL );
```

```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component
// All variables are data members of this class
```

```
_radiobox4 = XtVaCreateManagedWidget ( "radiobox4",
                                         xmRowColumnWidgetClass,
                                         _baseWidget,
                                         XmNOrientation, XmHORIZONTAL,
                                         XmNpacking, XmPACK_COLUMN,
                                         XmNradioBehavior, True,
                                         XmNradioAlwaysOne, True,
                                         XmNx, 58,
                                         XmNy, 115,
                                         XmNwidth, 153,
                                         XmNheight, 32,
                                         (XtPointer) NULL );
```

```
_toggleCombo = XtVaCreateManagedWidget ( "toggleCombo",
                                           xmToggleButtonWidgetClass,
                                           _radiobox4,
                                           XmNlabelType, XmSTRING,
                                           (XtPointer) NULL );
```

```
XtAddCallback ( _toggleCombo,
                XmNvalueChangedCallback,
                &BbDisplayUI::setToggleComboCallback,
                (XtPointer) this );
```

```
_toggleNormal = XtVaCreateManagedWidget ( "toggleNormal",
                                           xmToggleButtonWidgetClass,
                                           _radiobox4,
                                           XmNlabelType, XmSTRING,
                                           (XtPointer) NULL );
```

```
XtAddCallback ( _toggleNormal,
                XmNvalueChangedCallback,
                &BbDisplayUI::setToggleNormalCallback,
                (XtPointer) this );
```

```
_optionMenu3 = new VkOptionMenu ( _baseWidget, "optionMenu3");
_optionWhole2D = _optionMenu3->addAction ( "optionWhole2D",
                                           &BbDisplayUI::doOptionWhole2DCallback,
```

```

_optionL3D = _optionMenu3->addAction ( "optionL3D",
                                       &BbDisplayUI::doOptionL3DCallback,
                                       (XtPointer) this );

_optionFlow2D = _optionMenu3->addAction ( "optionFlow2D",
                                       &BbDisplayUI::doOptionFlow2DCallback,
                                       (XtPointer) this );

_optionFlow3D = _optionMenu3->addAction ( "optionFlow3D",
                                       &BbDisplayUI::doOptionFlow3DCallback,
                                       (XtPointer) this );

_radioboxZoom = XtVaCreateManagedWidget ( "radioboxZoom",
                                       xmRowColumnWidgetClass,
                                       _baseWidget,
                                       XmNorientation, XmHORIZONTAL,
                                       XmNpacking, XmPACK_COLUMN,
                                       XmNradioBehavior, True,
                                       XmNradioAlwaysOne, True,
                                       XmNx, 361,
                                       XmNy, 110,
                                       XmNwidth, 192,
                                       XmNheight, 32,
                                       (XtPointer) NULL );

_toggleLeft = XtVaCreateManagedWidget ( "toggleLeft",
                                       xmToggleButtonWidgetClass,
                                       _radioboxZoom,
                                       XmNlabelType, XmSTRING,
                                       (XtPointer) NULL );

XtAddCallback ( _toggleLeft,
               XmNvalueChangedCallback,
               &BbDisplayUI::setToggleLeftCallback,
               (XtPointer) this );

_toggleRight = XtVaCreateManagedWidget ( "toggleRight",
                                       xmToggleButtonWidgetClass,
                                       _radioboxZoom,
                                       XmNlabelType, XmSTRING,
                                       (XtPointer) NULL );

XtAddCallback ( _toggleRight,
               XmNvalueChangedCallback,
               &BbDisplayUI::setToggleRightCallback,
               (XtPointer) this );

_toggleBoth = XtVaCreateManagedWidget ( "toggleBoth",
                                       xmToggleButtonWidgetClass,
                                       _radioboxZoom,
                                       XmNlabelType, XmSTRING,
                                       (XtPointer) NULL );

XtAddCallback ( _toggleBoth,
               XmNvalueChangedCallback,
               &BbDisplayUI::setToggleBothCallback,
               (XtPointer) this );

_optionMenuZoom = new VkOptionMenu ( _baseWidget, "optionMenuZoom");
_optionNormal = _optionMenuZoom->addAction ( "optionNormal",

```

```
&BbDisplayUI::doOptionNormalCallback,  
(XtPointer) this );          266
```

```
_optionDouble = _optionMenuZoom->addAction ( "optionDouble",
&BbDisplayUI::doOptionDoubleCallback,
(XtPointer) this );
```

```
_optionHalf = _optionMenuZoom->addAction ( "optionHalf",
&BbDisplayUI::doOptionHalfCallback,
(XtPointer) this );
```

```
_labelZoom = XtVaCreateManagedWidget ( "labelZoom",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 360,
                                         XmNy, 22,
                                         XmNwidth, 43,
                                         XmNheight, 20,
                                         (XtPointer) NULL );
```

```
_separator2 = XtVaCreateManagedWidget ( "separator2",
                                         xmSeparatorWidgetClass,
                                         _baseWidget,
                                         XmNorientation, XmVERTICAL,
                                         XmNx, 270,
                                         XmNy, 12,
                                         XmNwidth, 20,
                                         XmNheight, 140,
                                         (XtPointer) NULL );
```

[illegible][illegible]

```
XtAddCallback ( _textfieldZoom,
                XmNactivateCallback,
                &BbDisplayUI::imgZoomCallback,
                (XtPointer) this );
```

[illegible]

YEAN

t (

e

used

ient
112


```

{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionDouble ( w, callData );
}

void BbDisplayUI::doOptionFlow2DCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionFlow2D ( w, callData );
}

void BbDisplayUI::doOptionFlow3DCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionFlow3D ( w, callData );
}

void BbDisplayUI::doOptionHalfCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionHalf ( w, callData );
}

void BbDisplayUI::doOptionL3DCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionL3D ( w, callData );
}

void BbDisplayUI::doOptionNormalCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionNormal ( w, callData );
}

void BbDisplayUI::doOptionWhole2DCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->doOptionWhole2D ( w, callData );
}

void BbDisplayUI::imgNumCallback ( Widget      w,
                                  XtPointer clientData,
                                  XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->imgNum ( w, callData );
}

void BbDisplayUI::imgZoomCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;

```

```

    obj->imgZoom ( w, callData );
}

void BbDisplayUI::setToggleBothCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->setToggleBoth ( w, callData );
}

void BbDisplayUI::setToggleComboCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->setToggleCombo ( w, callData );
}

void BbDisplayUI::setToggleLeftCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->setToggleLeft ( w, callData );
}

void BbDisplayUI::setToggleNormalCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->setToggleNormal ( w, callData );
}

void BbDisplayUI::setToggleRightCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbDisplayUI* obj = ( BbDisplayUI * ) clientData;
    obj->setToggleRight ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbDisplayUI::doOptionDouble ( Widget, XtPointer )
{
    // This virtual function is called from doOptionDoubleCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionFlow2D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFlow2DCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionFlow3D ( Widget, XtPointer )
{

```

```

    // This virtual function is called from doOptionFlow3DCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionHalf ( Widget, XtPointer )
{
    // This virtual function is called from doOptionHalfCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionL3D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionL3DCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionNormal ( Widget, XtPointer )
{
    // This virtual function is called from doOptionNormalCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::doOptionWhole2D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionWhole2DCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::imgNum ( Widget, XtPointer )
{
    // This virtual function is called from imgNumCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::imgZoom ( Widget, XtPointer )
{
    // This virtual function is called from imgZoomCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::setToggleBoth ( Widget, XtPointer )
{
    // This virtual function is called from setToggleBothCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::setToggleCombo ( Widget, XtPointer )
{
    // This virtual function is called from setToggleComboCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::setToggleLeft ( Widget, XtPointer )
{
    // This virtual function is called from setToggleLeftCallback.
    // This function is normally overridden by a derived class.
}

```

```
}

void BbDisplayUI::setToggleNormal ( Widget, XtPointer )
{
    // This virtual function is called from setToggleNormalCallback.
    // This function is normally overridden by a derived class.
}

void BbDisplayUI::setToggleRight ( Widget, XtPointer )
{
    // This virtual function is called from setToggleRightCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbFlow
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFlowUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbFlow.h"
#include <Vk/VkEZ.h>
#include <Sgm/ThumbWheel.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbFlowUI and are
// available as protected data members inherited by this class
//
// XmTextField          _textfieldMin
// XmTextField          _textfieldMax
// SgThumbWheel         _thumbwheelSemi
// VkOptionMenu *       _optionMenuFlowMethod
// VkMenuItem *         _optionManual
// VkMenuItem *         _optionSemiAuto
// VkMenuItem *         _optionAutoSnake
// VkMenuItem *         _optionAutoEdge
// VkMenuItem *         _optionAutoThresh
// XmLabel              _labelArea
// XmLabel              _labelMV
// XmLabel              _labelBSV
// XmLabel              _labelPSV
// XmLabel              _labelVFR
// XmPushButton         _buttonAcceptFlow
// XmTextField          _textfieldArea
// XmTextField          _textfieldMV
// XmTextField          _textfieldBSV
// XmTextField          _textfieldPSV
// XmTextField          _textfieldVFR
//
////////////////////////////////////

```

```
//---- Start editable code block: headers and declarations
```

```
#include "Utility.h"
#include "Utility_Widget.h"
#include "BbRROI.h"
#include "snake.h"
#include "GS_Points.h"
```

```
//---- End editable code block: headers and declarations
```

```
//---- BbFlow Constructor
```

```
BbFlow::BbFlow(const char *name, Widget parent) :
    BbFlowUI(name, parent)
{
    // This constructor calls BbFlowUI(parent, name)
    // which calls BbFlowUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built
```

```
//---- Start editable code block: BbFlow constructor
```

```
//---- End editable code block: BbFlow constructor
```

```
}    // End Constructor
```

```
BbFlow::BbFlow(const char *name) :
    BbFlowUI(name)
```

```
{
    // This constructor calls BbFlowUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used
```

```
//---- Start editable code block: BbFlow constructor 2
```

```
//---- End editable code block: BbFlow constructor 2
```

```
}    // End Constructor
```

```
BbFlow::~BbFlow()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.
```

```
//---- Start editable code block: BbFlow destructor
```

```
//---- End editable code block: BbFlow destructor
```

```
}    // End Destructor
```

```

const char * BbFlow::className() // classname
{
    return ("BbFlow");
} // End className()

void BbFlow::SemiFlow ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFlow SemiFlow

    SgThumbWheelCallbackStruct *cbs = (SgThumbWheelCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::SemiFlow is implemented:
    //::VkUnimplemented ( w, "BbFlow::SemiFlow" );

    int tmp;
    SgThumbWheelGetValue(w, &tmp);
    float m = (float) tmp;
    float m;

    _objMag -> msgsRight.flow_noiseLevel = int(percent);

    if(_objMag -> msgsRight.flowDir < 0)
        m = (_objMag -> _imgView2 -> _minFlow) * percent / 100.0;
    else if(_objMag -> msgsRight.flowDir > 0)
        m = (_objMag -> _imgView2 -> _maxFlow) * percent / 100.0;

    set_noiseLevel(int(percent), m);

/*
    float maxI = -float(tmp);

    if(_objMag -> _imgView2 -> _ROI != NULL &&
        _objMag -> _imgView2 -> _ROI -> _area != NULL)

        _objMag -> _imgView2 -> semiFlow(-3000.0, maxI,
            _objMag -> _imgView2 -> _ROI -> _area);

*/

    //----- End editable code block: BbFlow SemiFlow
} // End BbFlow::SemiFlow()

void BbFlow::SemiFlowChg ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFlow SemiFlowChg

    SgThumbWheelCallbackStruct *cbs = (SgThumbWheelCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::SemiFlowChg is implemented:
    //::VkUnimplemented ( w, "BbFlow::SemiFlowChg" );

    //int tmp;
    //SgThumbWheelGetValue(w, &tmp);
    //printf("SemiFlow Chg  %d \n", tmp);

    //----- End editable code block: BbFlow SemiFlowChg
}

```

```

void BbFlow::doButtonAcceptFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow doButtonAcceptFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::doButtonAcceptFlow is implemented
    //::VkUnimplemented ( w, "BbFlow::doButtonAcceptFlow" );

    if(_objMag -> msgsRight.flow_method == FLOW_AUTOSNAKE)
    {
        GS_Points *p0 = new GS_Points();

        int i, step;

        Points *p = &(_objMag -> _imgView2 -> _ROI -> _points_in_border);
        if(p -> _numPoints < 100) step = 1;
        else if(p -> _numPoints < 200) step = 2;
        else if(p -> _numPoints < 300) step = 3;
        else if(p -> _numPoints < 400) step = 4;
        else step = 5;

        for(i=0; i< p -> _numPoints; i += step)
            p0 -> add( p -> _points[i].x, p -> _points[i].y );

        int row = _objMag->_imgView2->get_height();
        int col = _objMag->_imgView2->get_width();
        //-----

        printf("\n Snake Initial Points ==> %d (step = %d)\n", p0 -> _numPoints, step);

        GS_Points *p1 = snake(row, col, fimg, p0);

        p -> clear();
        for(i=0; i< p1 -> _numPoints; i++)
            p -> add(p1 -> _points[i].x, p1 -> _points[i].y );
        printf("\n Snake Points generated ==> %d \n", p1 -> _numPoints);
    }

    if(_objMag -> msgsRight.roi_flow != NULL)
    {
        printf("!!! msgsRight.roi_flow != NULL \n");
        free_img(_objMag -> msgsRight.roi_flow);
        _objMag -> msgsRight.roi_flow = NULL;
    }

    if(_objMag -> msgsRight.roi_back != NULL)
    {
        printf("!!! msgsRight.roi_back != NULL \n");
        free_img(_objMag -> msgsRight.roi_back);
        _objMag -> msgsRight.roi_back = NULL;
    }

    if(_objMag -> msgsLeft.user == USER_NOVIES)
    {
        printf("!!! msgsLeft.user == USER_NOVIES \n");
        _objMag -> _imgView2 -> AcceptROI();
        printf("!!! msgsLeft.user == USER_NOVIES 2\n");
        if(_objMag -> _imgView2 -> _ROI == NULL)
            ((BbROI *) (_objMag -> _RROI)) -> modify("Flow");
    }
}

```



```

if(_objMag -> _imgView2 -> _ROI != NULL)
_objMag -> msgsRight.roi_flow = _objMag -> _imgView2 -> _ROI -> copyArea276
printf("!!! msgsLeft.user == USER_NOVIES 3\n");
}
else
{
printf("!!! msgsLeft.user != USER_NOVIES\n");
if(((BbRROI *) (_objMag -> _RROI)) -> modify("Flow") == 1)
{
_objMag -> _imgView2 -> AcceptROI();
_objMag -> msgsRight.roi_flow = _objMag -> _imgView2 -> _ROI -> copyArea();
}

if(((BbRROI *) (_objMag -> _RROI)) -> modify("Back") == 1)
{
_objMag -> _imgView2 -> AcceptROI();
_objMag -> msgsRight.roi_back = _objMag -> _imgView2 -> _ROI -> copyArea();
}

int i = _objMag->msgsRight.img_number - _objMag->msgsLoaded.img_start;
((BbRROI *) (_objMag->_RROI)) -> draw_AllROI(i);
}

_objMag -> update_flow();

//----- End editable code block: BbFlow doButtonAcceptFlow
} // End BbFlow::doButtonAcceptFlow()

void BbFlow::doOptionAutoEdge ( Widget w, XtPointer callData )
{
//----- Start editable code block: BbFlow doOptionAutoEdge

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbFlow::doOptionAutoEdge is implemented:
::VkUnimplemented ( w, "BbFlow::doOptionAutoEdge" );

//----- End editable code block: BbFlow doOptionAutoEdge
} // End BbFlow::doOptionAutoEdge()

void BbFlow::doOptionAutoSnake ( Widget w, XtPointer callData )
{
//----- Start editable code block: BbFlow doOptionAutoSnake

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbFlow::doOptionAutoSnake is implemented:
//::VkUnimplemented ( w, "BbFlow::doOptionAutoSnake" );

_objMag -> msgsRight.flow_method = FLOW_AUTOSNAKE;

//----- End editable code block: BbFlow doOptionAutoSnake
} // End BbFlow::doOptionAutoSnake()

```

```
void BbFlow::doOptionAutoThresh ( Widget w, XtPointer callData )
```

277

```
{
    //----- Start editable code block: BbFlow doOptionAutoThresh

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::doOptionAutoThresh is implemented
    ::VkUnimplemented ( w, "BbFlow::doOptionAutoThresh" );

    //----- End editable code block: BbFlow doOptionAutoThresh
}    // End BbFlow::doOptionAutoThresh()
```

```
void BbFlow::doOptionManual ( Widget w, XtPointer callData )
```

```
{
    //----- Start editable code block: BbFlow doOptionManual

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::doOptionManual is implemented:
    //::VkUnimplemented ( w, "BbFlow::doOptionManual" );

    _objMag -> msgsRight.flow_method = FLOW_MANUAL;
    //_objMag -> update_flow();

    //----- End editable code block: BbFlow doOptionManual
}    // End BbFlow::doOptionManual()
```

```
void BbFlow::doOptionSemiAuto ( Widget w, XtPointer callData )
```

```
{
    //----- Start editable code block: BbFlow doOptionSemiAuto

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::doOptionSemiAuto is implemented:
    //::VkUnimplemented ( w, "BbFlow::doOptionSemiAuto" );

    _objMag -> msgsRight.flow_method = FLOW_SEMIAUTO;
    //_objMag -> update_flow();

    //----- End editable code block: BbFlow doOptionSemiAuto
}    // End BbFlow::doOptionSemiAuto()
```

```
void BbFlow::maxFlow ( Widget w, XtPointer callData )
```

```
{
    //----- Start editable code block: BbFlow maxFlow

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::maxFlow is implemented:
    ::VkUnimplemented ( w, "BbFlow::maxFlow" );
}
```

```

//---- End editable code block: BbFlow maxFlow
} // End BbFlow::maxFlow()

void BbFlow::minFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow minFlow

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow::minFlow is implemented:

    ::VkUnimplemented ( w, "BbFlow::minFlow" );

    //---- End editable code block: BbFlow minFlow
} // End BbFlow::minFlow()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *BbFlow::CreateBbFlow( const char *name, Widget parent )
{
    VkComponent *obj = new BbFlow ( name, parent );
    return ( obj );
} // End CreateBbFlow

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *BbFlow::RegisterBbFlowInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is

```

```

// selected, the name of the member function as a string,
// the type of the single argument to this function, and an.
// optional argument indicating the class that defines this function.
// All member functions must have the form
//
//     void memberFunction ( Type );
//
// where "Type" is one of:
//     const char *      (Use XmRString)
//     Boolean           (Use XmRBoolean)
//     int               (Use XmRInt)
//     float             (Use XmRFloat)
//     No argument       (Use VkrNoArg or "NoArg")
//     A filename        (Use VkrFilename or "Filename")
//     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
//     A callback        (Use XmRCallback)

static InterfaceMap map[] = {
//----- Start editable code block: BbFlowUI resource table

    // { "resourceName", "setAttribute", XmRString},
//----- End editable code block: BbFlowUI resource table
    { NULL }, // MUST be NULL terminated
};

return map;
} // End RegisterBbFlowInterface()

//----- End of generated code

//----- Start editable code block: End of generated code

void BbFlow::set_noiseLevel(int noise, float noiseflow)
{
    char str[30];
    sprintf(str, "%d %%", noise);
    XmTextFieldSetString(_textfieldMin, str);

    sprintf(str, "%5.2f", noiseflow/10.0);
    XmTextFieldSetString(_textfieldMax, str);
}

//----- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbFlow3D
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFlow3DUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "BbFlow3D.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

```

```

extern void VkUnimplemented ( Widget, const char * );

```

```

////////////////////////////////////
// The following non-container elements are created by BbFlow3DUI and are
// available as protected data members inherited by this class
//
// VkOptionMenu *           _optionMenu12
// VkMenuItem *             _optionPoint
// VkMenuItem *             _optionSurface
// VkOptionMenu *           _optionMenu10
// VkMenuItem *             _optionAddCut
// VkMenuItem *             _optionAddObj
// XmLabel                  _labelFlow3DEnd
// XmTextField               _textfieldFlow3DEnd
// XmLabel                  _labelFlow3DStart
// XmTextField               _textfieldFlow3DStart
// XmPushButton              _buttonSaveFlow3D
// XmLabel                  _labelFlow3DHigh
// XmLabel                  _labelFlow3DLow
// XmTextField               _textfieldFlow3DHigh
// XmTextField               _textfieldFlow3DLow
// XmTextField               _textfield1
// XmPushButton              _buttonAccept1
// XmToggleButton           _toggleDisable
// XmToggleButton           _toggleEnable
//
////////////////////////////////////

```

```
//---- Start editable code block: headers and declaration
```

```
#include "Bb3DLocalizer"
#include "Utility.h"
#include "BbRROI.h"
#include "ImgAlloc.h"
#include <math.h>

#include <Inventor/Sb.h>
#include <Inventor/nodes/SoCamera.h>
#include <Inventor/nodes/SoCoordinate3.h>
#include <Inventor/nodes/SoGroup.h>
#include <Inventor/nodes/SoLightModel.h>
#include <Inventor/nodes/SoPerspectiveCamera.h>
#include <Inventor/nodes/SoPointSet.h>
#include <Inventor/nodes/SoTransform.h>
#include <Inventor/nodes/SoSphere.h>
#include <Inventor/nodes/SoMaterialBinding.h>
#include <Inventor/nodes/SoBaseColor.h>
#include <Inventor/nodes/SoDrawStyle.h>
#include <Inventor/nodes/SoMaterial.h>
#include <Inventor/Xt/SoXt.h>
#include <Inventor/nodes/SoIndexedLineSet.h>
#include <Inventor/nodes/SoIndexedFaceSet.h>
```

```
float _cutVertex[4][3];
int _FaceIndex[5] = {0, 1, 2, 3, -1};
int _FaceIndex1[4] = {0, 1, 2, -1};
int _FaceIndex2[8] = {0, 1, 2, -1,
                     0, 2, 3, -1};
int _FaceIndex3[12] = {0, 1, 4, -1,
                      1, 3, 4, -1,
                      1, 2, 3, -1};
int _FaceIndex430[16] = {0, 1, 5, -1,
                        1, 2, 5, -1,
                        2, 4, 5, -1,
                        2, 3, 4, -1 };
int _FaceIndex431[16] = {0, 1, 2, -1,
                        0, 2, 4, -1,
                        2, 3, 4, -1,
                        0, 5, 4, -1};
```

```
//---- End editable code block: headers and declarations
```

```
//---- BbFlow3D Constructor
```

```
BbFlow3D::BbFlow3D(const char *name, Widget parent) :
    BbFlow3DUI(name, parent)
{
    // This constructor calls BbFlow3DUI(parent, name)
    // which calls BbFlow3DUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbFlow3D constructor

    _pts = 0;

    //---- End editable code block: BbFlow3D constructor

}    // End Constructor
```

```

//::VkUnimplemented ( w, "BbFlow3D::doOptionAddCut" )
addCut();

//---- End editable code block: BbFlow3D doOptionAddCut
} // End BbFlow3D::doOptionAddCut()

void BbFlow3D::doOptionAddObj ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow3D doOptionAddObj
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFlow3D::doOptionAddObj is implemented:
    //::VkUnimplemented ( w, "BbFlow3D::doOptionAddObj" );

    //---- End editable code block: BbFlow3D doOptionAddObj
} // End BbFlow3D::doOptionAddObj()

void BbFlow3D::doOptionPoint ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow3D doOptionPoint
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFlow3D::doOptionPoint is implemented:
    //::VkUnimplemented ( w, "BbFlow3D::doOptionPoint" );
    pointMaker(w);

    //---- End editable code block: BbFlow3D doOptionPoint
} // End BbFlow3D::doOptionPoint()

void BbFlow3D::doOptionSurface ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow3D doOptionSurface
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFlow3D::doOptionSurface is implemented:
    //::VkUnimplemented ( w, "BbFlow3D::doOptionSurface" );
    surfaceMaker(w);

    //---- End editable code block: BbFlow3D doOptionSurface
} // End BbFlow3D::doOptionSurface()

void BbFlow3D::setToggleDisable ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow3D setToggleDisable

```

```

XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
//--- Comment out the following line when BbFlow3D::setToggleDisable is implemented
::VkUnimplemented ( w, "BbFlow3D::setToggleDisable" );
_objMag -> msgsRight.flow3D = FLOW3D_DISABLE;

//---- End editable code block: BbFlow3D setToggleDisable
} // End BbFlow3D::setToggleDisable()

void BbFlow3D::setToggleEnable ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFlow3D setToggleEnable

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFlow3D::setToggleEnable is implemented:
    ::VkUnimplemented ( w, "BbFlow3D::setToggleEnable" );

    _objMag -> msgsRight.flow3D = FLOW3D_ENABLE;

    //---- End editable code block: BbFlow3D setToggleEnable
} // End BbFlow3D::setToggleEnable()

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

VkComponent *BbFlow3D::CreateBbFlow3D( const char *name, Widget parent )
{
    VkComponent *obj = new BbFlow3D ( name, parent );
    return ( obj );
} // End CreateBbFlow3D

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```



```
void *BbFlow3D::RegisterBbFlow3DInterface()
```

284

```
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //      void memberFunction ( Type );
    //
    // where "Type" is one of:
    //      const char *      (Use XmRString)
    //      Boolean           (Use XmRBoolean)
    //      int                (Use XmRInt)
    //      float             (Use XmRFloat)
    //      No argument       (Use VkRNoArg or "NoArg")
    //      A filename        (Use VkRFilename or "Filename")
    //      An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //      A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbFlow3DUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbFlow3DUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbFlow3DInterface()
```

```
//---- End of generated code
```

```
//---- Start editable code block: End of generated code
```

```
int BbFlow3D::getJointPoint_2LinesIn3DSpace(float x1, float y1, float z1, float p1, float q1, float r1,
float x2, float y2, float z2, float p2, float q2, float r2, float *x, float *y, float *z)
{
    float t2;

    if(fabsf(p1*q2 - q1*p2) > 1.e-20)
        t2 = ( (p1*y1-q1*x1)-(p1*y2-q1*x2) ) / (p1*q2 - q1*p2);
    else if(fabsf(p1*r2 - r1*p2) > 1.e-20)
        t2 = ( (p1*z1-r1*x1)-(p1*z2-r1*x2) ) / (p1*r2 - r1*p2);
    else if(fabsf(q1*r2 - r1*q2) > 1.e-20)
        t2 = ( (q1*z1-r1*y1)-(q1*z2-r1*y2) ) / (q1*r2 - r1*q2);
    else
        return 0;

    *x = x2 + p2*t2;
    *y = y2 + q2*t2;
    *z = z2 + r2*t2;
    return 1;
}

void BbFlow3D::get_axialPlaneVertex(int number)
{

```

```

int i = 0;
int j = 0;
_planeVertex[0][0] = (_tlh_R - j * _pX);
_planeVertex[0][2] = _tlh_S + _thick * number;
_planeVertex[0][1] = _tlh_A - i * _pY;
i = 0;
j = _width - 1;
_planeVertex[1][0] = (_tlh_R - j * _pX);
_planeVertex[1][2] = _tlh_S + _thick * number;
_planeVertex[1][1] = _tlh_A - i * _pY;
i = _height - 1;
j = _width - 1;
_planeVertex[2][0] = (_tlh_R - j * _pX);
_planeVertex[2][2] = _tlh_S + _thick * number;
_planeVertex[2][1] = _tlh_A - i * _pY;
i = _height - 1;
j = 0;
_planeVertex[3][0] = (_tlh_R - j * _pX);
_planeVertex[3][2] = _tlh_S + _thick * number;
_planeVertex[3][1] = _tlh_A - i * _pY;
}

```

```

void BbFlow3D::get_sagittalPlaneVertex(int number)
{

```

```

    int j = number;

    int k = 0;

    int i = 0;
    _planeVertex[0][2] = _tlh_S + _thick * k;
    _planeVertex[0][1] = _tlh_A - i * _pY;
    i = _height - 1;
    _planeVertex[1][0] = (_tlh_R - j * _pX);
    _planeVertex[1][2] = _tlh_S + _thick * k;
    _planeVertex[1][1] = _tlh_A - i * _pY;

    k = _depth - 1;

    i = 0;
    _planeVertex[3][2] = _tlh_S + _thick * k;
    _planeVertex[3][1] = _tlh_A - i * _pY;
    i = _height - 1;
    _planeVertex[2][0] = (_tlh_R - j * _pX);
    _planeVertex[2][2] = _tlh_S + _thick * k;
    _planeVertex[2][1] = _tlh_A - i * _pY;
}

```

```

void BbFlow3D::get_coronalPlaneVertex(int number)
{

```

```

    int i = number;

    int k = 0;

    int j = 0;
    _planeVertex[0][2] = _tlh_S + _thick * k;
    _planeVertex[0][1] = _tlh_A - i * _pY;
    j = _width - 1;
    _planeVertex[1][0] = (_tlh_R - j * _pX);
    _planeVertex[1][2] = _tlh_S + _thick * k;
    _planeVertex[1][1] = _tlh_A - i * _pY;

    k = _depth - 1;
}

```

```

    j = 0;
    _planeVertex[3][0] = (_tlh_R - j * _pX);
    _planeVertex[3][2] = _tlh_S + _thick * k;
    _planeVertex[3][1] = _tlh_A - i * _pY;
    j = _width - 1;
    _planeVertex[2][0] = -(_tlh_R - j * _pX);
    _planeVertex[2][2] = _tlh_S + _thick * k;
    _planeVertex[2][1] = _tlh_A - i * _pY;
}

ImgGE *BbFlow3D::get_axialPlaneImg(int number)
{
    ImgGE *imgGE = new ImgGE();

    short **img = alloc_shimg(_width, _height);

    for(int i=0; i<_height; i++)
        for(int j=0; j<_width; j++)
            img[i][j] = _data3d[j][i][number];

    imgGE -> set_width(_width);
    imgGE -> set_height(_height);
    imgGE -> set_imgdata(img);

    return imgGE;
}

ImgGE *BbFlow3D::get_sagittalPlaneImg(int number)
{
    ImgGE *imgGE = new ImgGE();

    printf("\n ****    number = %d    %d    %d ***\n", number, _height, _depth);

    short **img = alloc_shimg(_height, _depth);

    for(int i=0; i<_depth; i++)
        for(int j=0; j<_height; j++)
            img[i][j] = _data3d[number][j][i];

    imgGE -> set_width(_height);
    imgGE -> set_height(_depth);
    imgGE -> set_imgdata(img);

    return imgGE;
}

ImgGE *BbFlow3D::get_coronalPlaneImg(int number)
{
    ImgGE *imgGE = new ImgGE();

    short **img = alloc_shimg(_width, _depth);

    for(int i=0; i<_depth; i++)
        for(int j=0; j<_width; j++)
            img[i][j] = _data3d[j][number][i];

    imgGE -> set_width(_width);
    imgGE -> set_height(_depth);
    imgGE -> set_imgdata(img);

    return imgGE;
}

void BbFlow3D::get_axial(int number, float x1, float y1,

```

```

    float *x, float *y, float *z)
{
    *x = (_tlh_R - x1 * _pX);
    *z = _tlh_S + _thick * number;
    *y = _tlh_A - y1 * _pY;
}

void BbFlow3D::get_sagital(int number, float x1, float y1,
    float *x, float *y, float *z)
{
    *x = (_tlh_R - number * _pX);
    *z = _tlh_S + _thick * y1;
    *y = _tlh_A - x1 * _pY;
}

void BbFlow3D::get_coronal(int number, float x1, float y1,
    float *x, float *y, float *z)
{
    *x = (_tlh_R - x1 * _pX);
    *z = _tlh_S + _thick * y1;
    *y = _tlh_A - number * _pY;
}

void BbFlow3D::get_transform(float x1, float y1, float *x2, float *y2, float *x, float
{
    *x2 = x1 * _pX;
    *y2 = y1 * _pY;
    *x = _cutVertex[0][0];
    *y = _cutVertex[0][1];
    *z = _cutVertex[0][2];
}

void BbFlow3D::get_anyCut(float *p1, float *p2, float *p3,
    float *q1, float *q2, float *q3, float *r1, float *r2, float *r3)
{
    float dx, dy, dz, d;

    dx = _cutVertex[1][0] - _cutVertex[0][0];
    dy = _cutVertex[1][1] - _cutVertex[0][1];
    dz = _cutVertex[1][2] - _cutVertex[0][2];
    d = sqrt(dx*dx+dy*dy+dz*dz);
    *p1 = dx/d;
    *p2 = dy/d;
    *p3 = dz/d;

    // [REDACTED]
    dz = _cutVertex[3][2] - _cutVertex[0][2];
    d = sqrt(dx*dx+dy*dy+dz*dz);
    *q1 = dx/d;
    *q2 = dy/d;
    *q3 = dz/d;

    *r1 = (*p2) * (*q3) - (*p3) * (*q2);
    *r2 = (*p3) * (*q1) - (*p1) * (*q3);
    *r3 = (*p1) * (*q2) - (*p2) * (*q1);
}

void BbFlow3D::get_axialCenter(int number, unsigned char **area,
    float *x, float *y, float *z)
{
    float x1 = 0;
    float y1 = 0;
    float w = 0;
    float tmp;

```

```

for(int i=0; i<_height; i++)
for(int j=0; j<_width; j++)
{
    tmp = _data3d[j][i][number];
    if(area[i][j] == 1 && tmp > _lowThreshold &&
        tmp < _highThreshold)
    {
        tmp = 1;
        x1 += float(j) * tmp;
        y1 += float(i) * tmp;
        w += tmp;
    }
}

if(w != 0)
{
    x1 /= w;
    y1 /= w;
}

*x = (_tlh_R - x1 * _pX);
*z = _tlh_S + _thick * number;
*y = _tlh_A - y1 * _pY;
}

void BbFlow3D::get_sagittalCenter(int number, unsigned char **area,
    float *x, float *y, float *z)
{
    float x1 = 0;
    float y1 = 0;
    float w = 0;
    float tmp;

    /*
     * Determine the center point of the cut area by averging approach
     */
    /*
    for(int i=0; i<_depth; i++)
    for(int j=0; j<_height; j++)
    {
        tmp = _data3d[number][j][i];
        if(area[i][j] == 1 && tmp > _lowThreshold &&
            tmp < _highThreshold)
        {
            tmp = 1;
            x1 += float(j) * tmp;
            y1 += float(i) * tmp;
            w += tmp;
        }
    }

    if(w != 0)
    {
        x1 /= w;
        y1 /= w;
    }
    */

    /*
     * Determine the center point of the cut area by the highest intensity
     */

    float maxI = -1.e20;
    for(int i=0; i<_depth; i++)

```

```

for(int j=0; j<_height; j++)
{
    tmp = _data3d[number][j][i];
    if(area[i][j] == 1 && tmp > _lowThreshold &&
        tmp < _highThreshold)
    {
        if(tmp > maxI) {maxI = tmp; y1 = i; x1 = j;}
    }
}

printf(" $$$ x1 = %f y1 = %f\n", x1, y1);

*x = (_tlh_R - number * _pX);
*z = _tlh_S + _thick * y1;
*y = _tlh_A - x1 * _pY;
}

void BbFlow3D::get_coronalCenter(int number, unsigned char **area,
    float *x, float *y, float *z)
{
    float x1 = 0;
    float y1 = 0;
    float w = 0;
    float tmp;

    for(int i=0; i<_depth; i++)
    for(int j=0; j<_width; j++)
    {
        tmp = _data3d[j][number][i];
        if(area[i][j] == 1 && tmp > _lowThreshold &&
            tmp < _highThreshold)
        {
            tmp = 1;
            x1 += float(j) * tmp;
            y1 += float(i) * tmp;
            w += tmp;
        }
    }

    if(w != 0)
    {
        x1 /= w;
        y1 /= w;
    }

    *x = (_tlh_R - x1 * _pX);
    *z = _tlh_S + _thick * y1;
    *y = _tlh_A - number * _pY;
}

void BbFlow3D::pointMaker(Widget wid)
{
    if(_pts != 0)
    {
        pointMaker0();
        sceneMaker(wid);

        SoSeparator *root = (SoSeparator *) (_objMag->_root);
        SoSeparator *obj = (SoSeparator *) (root->getChild(2));
        SoSeparator *vessels = (SoSeparator *) (obj->getChild(0));

        SoTransform *velTransform = new SoTransform;
        vessels -> addChild(velTransform);

        SoMaterialBinding *velMaterialBinding = new SoMaterialBinding;
        vessels -> addChild(velMaterialBinding);
    }
}

```

```

//
SoXt::init(wid);

if(_objMag -> _root != NULL)
{
    _objMag -> _root -> unref();
    _objMag -> _root = NULL;
}

SoSeparator *root = new SoSeparator;
root -> ref();
_objMag -> _root = (SoSeparator *)root;

SoPerspectiveCamera *myCamera = new SoPerspectiveCamera;
root->addChild(myCamera);

SoTransform *myTransform = new SoTransform;
root->addChild(myTransform);

SoSeparator *obj = new SoSeparator;
root->addChild(obj);

//
// set up myCamera
//

float md;
if( (_xhigh - _xlow) > (_yhigh - _ylow) ) md = _xhigh - _xlow;
else md = _yhigh - _ylow;
if( md < (_zhigh - _zlow) ) md = _zhigh - _zlow;

myCamera->position.setValue( _xc, _yc, _zc + 15.0*md );
myCamera->nearDistance.setValue( 2.0*md );
myCamera->farDistance.setValue( 28.0*md );

//
// set up myTransform
//
myTransform->center.setValue(_xc, _yc, _zc);

//
// set up obj
//
SoSeparator *vessels = new SoSeparator;
obj->addChild(vessels);

SoSeparator *ball = new SoSeparator;
obj->addChild(ball);

SoSeparator *box = new SoSeparator;
obj -> addChild(box);

SoSeparator *axes = new SoSeparator;
obj -> addChild(axes);

SoSeparator *plane = new SoSeparator;
obj -> addChild(plane);

SoSeparator *redBall = new SoSeparator;
obj -> addChild(redBall);

SoSeparator *yellowBall = new SoSeparator;
obj -> addChild(yellowBall);

```

```

SoSeparator *roi = new SoSeparator;
obj -> addChild(roi);

SoSeparator *cline = new SoSeparator;
obj -> addChild(cline);

SoSeparator *sline = new SoSeparator;
obj -> addChild(sline);

SoSeparator *cut = new SoSeparator;
obj -> addChild(cut);

//
// set up Red Ball
//
SoTransform *ballTransform0 = new SoTransform;
redBall->addChild(ballTransform0);

SoMaterial *ballMaterial0 = new SoMaterial;
redBall -> addChild(ballMaterial0);

SoSphere *ballSphere0 = new SoSphere;
redBall -> addChild(ballSphere0);

ballMaterial0->emissiveColor.setValue(1, 0, 0);

//
// set up Yellow Ball
//
SoTransform *ballTransform1 = new SoTransform;
yellowBall->addChild(ballTransform1);

SoMaterial *ballMaterial1 = new SoMaterial;
yellowBall -> addChild(ballMaterial1);

SoSphere *ballSphere1 = new SoSphere;
yellowBall -> addChild(ballSphere1);

ballMaterial1->emissiveColor.setValue(1, 1, 0);

//
// set up ball
//
SoTransform *ballTransform = new SoTransform;
ball->addChild(ballTransform);

SoMaterial *ballMaterial = new SoMaterial;
ball -> addChild(ballMaterial);

SoSphere *ballSphere = new SoSphere;
ball -> addChild(ballSphere);

ballTransform->translation.setValue(_xc, _yc, _zc);
ballTransform->center.setValue(_xc, _yc, _zc);
//ballMaterial->ambientColor.setValue(0.9, 0., 0.);
//ballMaterial->diffuseColor.setValue(1, 0., 0);
ballMaterial->emissiveColor.setValue(0, 1, 0);
//ballMaterial->transparency.setValue(0.5);
ballSphere -> radius.setValue(2.0);

//
// set up vessels
//

//

```



```

// set up box
//
SoMaterial *boxMaterial = new SoMaterial;
box -> addChild(boxMaterial);

SoCoordinate3 *boxCoord = new SoCoordinate3;
box -> addChild(boxCoord);

SoIndexedLineSet *boxLine = new SoIndexedLineSet;
box -> addChild(boxLine);

boxMaterial->diffuseColor.setValue(1, 0., 0);
boxMaterial->emissiveColor.setValue(0.3, 0., 0);
boxMaterial->transparency.setValue(0.5);
boxCoord->point.setValues(0, 8, _box);
boxLine->coordIndex.setValues(0, 36, _boxLineIndex);

//
// set up axes
//
SoSeparator *axisR = new SoSeparator;
axes -> addChild(axisR);

SoSeparator *axisA = new SoSeparator;
axes -> addChild(axisA);

SoSeparator *axisS = new SoSeparator;
axes -> addChild(axisS);

int _LineIndex[3];
_LineIndex[0] = 0;
_LineIndex[1] = 1;
_LineIndex[2] = -1;

float _linePoint[2][3];
_linePoint[0][0] = _xc;
_linePoint[0][1] = _yc;
_linePoint[0][2] = _zc;

//
// set up axisR
//
_linePoint[1][0] = _xc + (_xhigh-_xlow)/2;
_linePoint[1][1] = _yc;
_linePoint[1][2] = _zc;

SoMaterial *rMaterial = new SoMaterial;
axisR -> addChild(rMaterial);

SoCoordinate3 *rCoord = new SoCoordinate3;
axisR -> addChild(rCoord);

SoIndexedLineSet *rLine = new SoIndexedLineSet;
axisR -> addChild(rLine);

rMaterial->emissiveColor.setValue(1, 0., 0);
rMaterial->diffuseColor.setValue(1, 0., 0);
rCoord->point.setValues(0, 2, _linePoint);
rLine->coordIndex.setValues(0, 3, _LineIndex);

//
// set up axisA
//
_linePoint[1][0] = _xc;
_linePoint[1][1] = _yc + (_yhigh - _ylow)/2.0;
_linePoint[1][2] = _zc;

```

```

SoMaterial *aMaterial = new SoMaterial;
axisA -> addChild(aMaterial);

SoCoordinate3 *aCoord = new SoCoordinate3;
axisA -> addChild(aCoord);

SoIndexedLineSet *aLine = new SoIndexedLineSet;
axisA -> addChild(aLine);

aMaterial->emissiveColor.setValue(0, 1., 0);
aMaterial->diffuseColor.setValue(0, 1., 0);
aCoord->point.setValues(0, 2, _linePoint);
aLine->coordIndex.setValues(0, 3, _LineIndex);

//
// set up axisS
//
_linePoint[1][0] = _xc;
_linePoint[1][1] = _yc;
_linePoint[1][2] = _zc + (_zhigh - _zlow)/2;

SoMaterial *sMaterial = new SoMaterial;
axisS -> addChild(sMaterial);

SoCoordinate3 *sCoord = new SoCoordinate3;
axisS -> addChild(sCoord);

SoIndexedLineSet *sLine = new SoIndexedLineSet;
axisS -> addChild(sLine);

sMaterial->emissiveColor.setValue(0, 0., 1);
sMaterial->diffuseColor.setValue(0, 0., 1);
sCoord->point.setValues(0, 2, _linePoint);
sLine->coordIndex.setValues(0, 3, _LineIndex);

//
// set up plane
//

SoMaterial *pMaterial = new SoMaterial;
plane -> addChild(pMaterial);

SoCoordinate3 *pCoord = new SoCoordinate3;
plane -> addChild(pCoord);

SoIndexedFaceSet *pFace = new SoIndexedFaceSet;
plane -> addChild(pFace);

//pCoord->point.setValues(0, 4, _planeVertex);
//pFace->coordIndex.setValues(0, 5, _FaceIndex);

//
// set up ROI
//

SoMaterial *roiMaterial = new SoMaterial;
roi -> addChild(roiMaterial);

SoCoordinate3 *roiCoord = new SoCoordinate3;
roi -> addChild(roiCoord);

```

```

SoIndexedFaceSet *roiFace = new SoIndexedFaceSet;
roi -> addChild(roiFace);

roiMaterial->emissiveColor.setValue(1, 0, 0.);
roiMaterial->diffuseColor.setValue(1, 0, 0.);
roiMaterial->transparency.setValue(0.5);

//roiCoord->point.setValues(0, n, _roiVertex);
//roiFace->coordIndex.setValues(0, n+1, _roiFaceIndex);

//
// set up central line
//
SoMaterial *clMaterial = new SoMaterial;
cline -> addChild(clMaterial);

SoCoordinate3 *clCoord = new SoCoordinate3;
cline -> addChild(clCoord);

SoIndexedLineSet *clLine = new SoIndexedLineSet;
cline -> addChild(clLine);

clMaterial->diffuseColor.setValue(1, 0., 0);
clMaterial->emissiveColor.setValue(1, 0., 0);
clMaterial->transparency.setValue(0.5);
//clCoord->point.setValues(0, ?, _cline);
//clLine->coordIndex.setValues(0, ?, _clLineIndex);

//
// set up straight line
//
SoMaterial *slMaterial = new SoMaterial;
sline -> addChild(slMaterial);

SoCoordinate3 *slCoord = new SoCoordinate3;
sline -> addChild(slCoord);

SoIndexedLineSet *slLine = new SoIndexedLineSet;
sline -> addChild(slLine);

slMaterial->diffuseColor.setValue(0, 1., 0);
slMaterial->emissiveColor.setValue(0, 1., 0);
slMaterial->transparency.setValue(0.5);

//
// set up cut
//

SoMaterial *cutMaterial = new SoMaterial;
cut -> addChild(cutMaterial);

SoCoordinate3 *cutCoord = new SoCoordinate3;
cut -> addChild(cutCoord);

SoIndexedFaceSet *cutFace = new SoIndexedFaceSet;
cut -> addChild(cutFace);

cutMaterial->emissiveColor.setValue(0, 1., 1.0);
cutMaterial->diffuseColor.setValue(0, 1., 1.0);
cutMaterial->transparency.setValue(0.5);

/*
root->unrefNoDelete();

if(_objMag -> _root != NULL)
{

```

```

    _objMag -> _root -> unref();
    _objMag -> _root = NULL;
}
_objMag -> _root = (SoSeparator *)root;
*/

XmTextFieldSetString(_textfield1, "Save is done !");

}

void BbFlow3D::addCut()
{
    GE_PCMRA_HEADER_OBJ *pc = _objMag -> _GE_header;
    float x1, y1, z1, p1, q1, r1, x2, y2, z2, p2, q2, r2;
    float x3, y3, z3, x4, y4, z4, d;

    _pX = pc->pixsize_X;
    _pY = pc->pixsize_Y;

    x1 = pc->tlh_R;
    y1 = pc->tlh_A;
    z1 = pc->tlh_S;
    x2 = pc->brh_R;
    y2 = pc->brh_A;
    z2 = pc->brh_S;
    x3 = pc->trh_R;
    y3 = pc->trh_A;
    z3 = pc->trh_S;

    printf(" BbFlow3D::addCut  %f %f %f    %f %f %f    %f %f %f\n", x1, y1, z1,
        x2, y2, z2, x3, y3, z3);

    d = fsqrt((x2-x3)*(x2-x3) + (y2-y3)*(y2-y3) + (z2-z3)*(z2-z3));
    p1 = (x2-x3)/d;
    q1 = (y2-y3)/d;
    r1 = (z2-z3)/d;
    d = fsqrt((x1-x3)*(x1-x3) + (y1-y3)*(y1-y3) + (z1-z3)*(z1-z3));
    p2 = (x1-x3)/d;
    q2 = (y1-y3)/d;
    r2 = (z1-z3)/d;

    if(getJointPoint_2LinesIn3DSpace(x1, y1, z1, p1, q1, r1, x2, y2, z2, p2, q2, r2, &*)
    {
        if(_objMag -> _root != NULL)
        {
            ((Bb3DLocalizer *) (_objMag->_localizer3d)) -> set_whichcut(_CUT_ANY);

            /*sometime gives blue cut
            SoSeparator *root = _objMag -> _root;
            SoSeparator *obj = (SoSeparator *) (root->getChild(2));
            SoSeparator *cut = (SoSeparator *) (obj -> getChild(10));
            */

            /*
            SoSeparator *cut = new SoSeparator;
            obj -> addChild(cut);
            SoMaterial *cutMaterial = new SoMaterial;
            cut -> addChild(cutMaterial);
            SoCoordinate3 *cutCoord = new SoCoordinate3;
            cut -> addChild(cutCoord);
            SoIndexedFaceSet *cutFace = new SoIndexedFaceSet;
            cut -> addChild(cutFace);
            */

            /*sometime gives blue cut

```

```

    fprintf(fp, "                LightModel {\n");
    fprintf(fp, "                model BASE_COLOR\n");
    fprintf(fp, "                }\n");

    fprintf(fp, "                BaseColor { \n");
    fprintf(fp, "                rgb [\n");

    fprintf(fp, "        1 0 0, \n");
    fprintf(fp, "        0 1 0, \n");
    fprintf(fp, "        0 0 1, \n");
    fprintf(fp, "        1 1 0, \n");

    fprintf(fp, "                ]\n");
    fprintf(fp, "                }\n");

    fprintf(fp, "                DrawStyle {\n");
    fprintf(fp, "                style POINTS\n");
    fprintf(fp, "                pointSize 10\n");
    fprintf(fp, "                }\n");

    fprintf(fp, "                USE CutCoords\n");

    fprintf(fp, "                PointSet {\n");
    fprintf(fp, "                startIndex    0\n");
    fprintf(fp, "                numPoints    4\n");
    fprintf(fp, "                }\n");

    fprintf(fp, "    } \n");

    fprintf(fp, ")\n");
    fclose(fp);

    XmTextFieldSetString(_textfield1, "Save is done !");
}
else
    XmTextFieldSetString(_textfield1, "No Points");
*/
}
}

void BbFlow3D::surfaceMaker(Widget wid)
{
    surfaceMaker0();
    sceneMaker(wid);

    SoSeparator *root = (SoSeparator *) (_objMag->_root);
    SoSeparator *obj = (SoSeparator *) (root->getChild(2));
    SoSeparator *vessels = (SoSeparator *) (obj->getChild(0));

    SoTransform *velTransform = new SoTransform;
    vessels -> addChild(velTransform);

    SoMaterial *velMaterial = new SoMaterial;
    vessels -> addChild(velMaterial);

    velMaterial->ambientColor.setValue(0.2, 0.2, 0.2);
    velMaterial->diffuseColor.setValue(0.2, 0.2, 0.2);
    velMaterial->emissiveColor.setValue(1, 0.2, 0.2);
    velMaterial->transparency.setValue(0.5);

    velTransform -> center.setValue(_xc, _yc, _zc);

    _isoThreshold = _lowThreshold;

    connect[0][0] = 1;

```

```

    }
}

set_scene();
}

void BbFlow3D::surfaceMaker0()
{
/*
    int x = 0;
    int y = _height - 1;
    int z = 0;
    _xlow = (_tlh_R - x * _pX);
    _ylow = _tlh_A - y * _pY;
    _zlow = _tlh_S + _thick * z;

    x = _width - 1;
    y = 0;
    z = _depth - 1;
    _xhigh = (_tlh_R - x * _pX);
    _yhigh = _tlh_A - y * _pY;
    _zhigh = _tlh_S + _thick * z;

    _xc = (_xlow + _xhigh)/2.0;
    _yc = (_ylow + _yhigh)/2.0;
    _zc = (_zlow + _zhigh)/2.0;
*/
    if(_pts != 0)
    {
        int i;

        _low = _xlow = _ylow = _zlow = 1.e30;
        _high = _xhigh = _yhigh = _zhigh = -1.e30;

        for(i=0; i<_pts; i++)
        {
            if(_vertex[i][0] > _xhigh) _xhigh = _vertex[i][0];
            if(_vertex[i][0] < _xlow) _xlow = _vertex[i][0];
            if(_vertex[i][1] > _yhigh) _yhigh = _vertex[i][1];
            if(_vertex[i][1] < _ylow) _ylow = _vertex[i][1];
            if(_vertex[i][2] > _zhigh) _zhigh = _vertex[i][2];
            if(_vertex[i][2] < _zlow) _zlow = _vertex[i][2];
            if(_rgb[i][0] > _high) _high = _rgb[i][0];
            if(_rgb[i][0] < _low) _low = _rgb[i][0];
        }

        _xc = (_xlow + _xhigh)/2.0;
        _yc = (_ylow + _yhigh)/2.0;
        _zc = (_zlow + _zhigh)/2.0;
    }
}

void BbFlow3D::marchingCube()
{
    int i, im1, im2, im3, im4, tag, flag2[8];

    int num = 0;
    int num2 = 0;

    for(i=0; i<8; i++)
    {
        if(_data3d[p[i][0]][p[i][1]][p[i][2]] >= _isoThreshold)
        {
            flag[num] = i;
            ++num;
        }
    }
}

```

```

        break;
    default:
        break;
}
/*
if(num == 3 && tag == 0)
{
    printf(" num = %d      tag = %d \n", num, tag);
    getchar();
}
if(num == 4 && (tag == 21 || tag == 0))
{
    printf(" num = %d      tag = %d \n", num, tag);
    getchar();
}
*/
}

void BbFlow3D::marchingCube1(int im)
{
    int i;
    float mid[3];

    if(_data3d[p[im][0]][p[im][1]][p[im][2]] != _isoThreshold)
    {
        for(i=0; i<3; i++)
        {
            [REDACTED]
        }
        add_to_scene(1);
    }
}

void BbFlow3D::marchingCube2(int im1, int im2)
{
    int i, k1, k2;
    int p1[2], p2[2];
    float mid[3];

    k1 = k2 = 0;
    for(i=0; i<3; i++)
    {
        if(connect[im1][i] != im2)
        {
            p1[k1++] = connect[im1][i];
        }
        if(connect[im2][i] != im1)
        {
            p2[k2++] = connect[im2][i];
        }
    }
    for(i=0; i<2; i++)
    {
        if(neighbor2(p1[i], p2[0])) k1 = p1[i];
        else k2 = p1[i];
    }
    [REDACTED]
    interpolatePoint(im1, k2, mid);
    add_res(mid, 3);

    interpolatePoint(im1, p2[0], mid);
    add_res(mid, 1);
    interpolatePoint(im1, p2[1], mid);

```

```

    add_res(mid,2);
    add_to_scene(2);
}

void BbFlow3D::marchingCube3(int im1, int im2, int im3)
{
    int i, k1, k2, k3;
    int p2[2], p3[2];
    float mid[3];

    k2 = k3 = 0;
    for(i=0; i<3; i++)
    {
        if(connect[im1][i] != im2 && connect[im1][i] != im3)
        {
            k1 = connect[im1][i];
        }
        if(connect[im2][i] != im1)
        {
            p2[k2++] = connect[im2][i];
        }
        if(connect[im3][i] != im1)
        {
            p3[k3++] = connect[im3][i];
        }
    }

    for(i=0; i<2; i++)
    {
        if(neighbor2(k1,p2[i])) k2 = p2[i];
        else k3 = p2[i];
    }

    interpolatePoint(im1, k1, mid);
    add_res(mid,0);
    interpolatePoint(im2, k2, mid);
    add_res(mid,1);
    interpolatePoint(im2, k3, mid);
    add_res(mid,2);

    for(i=0; i<2; i++)
    {
        if(neighbor2(k1,p3[i])) k2 = p3[i];
        else k3 = p3[i];
    }
    interpolatePoint(im3, k3, mid);
    add_res(mid,3);
    interpolatePoint(im3, k2, mid);
    add_res(mid,4);

    add_to_scene(3);
}

void BbFlow3D::marchingCube44(int im1, int im2, int im3, int im4)
{
    int i, p1[4], p2[4];
    float mid[3];

    p1[0] = im1;
    if(neighbor2(im1, im2))
    {
        p1[1] = im2;
        if(neighbor2(im1, im3)) {p1[3] = im3; p1[2] = im4;}
        else {p1[2] = im3; p1[3] = im4;}
    }
}

```



```

else
{
    p1[2] = im2;
    p1[1] = im3;
    p1[3] = im4;
}

for(i=0; i<3; i++)
{
    if(connect[p1[0]][i] != p1[1] && connect[p1[0]][i] != p1[3])
    {
        p2[0] = connect[p1[0]][i];
    }
    if(connect[p1[1]][i] != p1[0] && connect[p1[1]][i] != p1[2])
    {
        p2[1] = connect[p1[1]][i];
    }
    if(connect[p1[2]][i] != p1[1] && connect[p1[2]][i] != p1[3])
    {
        p2[2] = connect[p1[2]][i];
    }
    if(connect[p1[3]][i] != p1[0] && connect[p1[3]][i] != p1[2])
    {
        p2[3] = connect[p1[3]][i];
    }
}

for(i=0; i<4; i++)
{
    interpolatePoint(p1[i], p2[i], mid);
    add_res(mid,i);
}

add_to_scene(44);
}

void BbFlow3D::marchingCube430(int im1, int im2, int im3, int im4)
{
    int i, k2, k3, k4;
    int p2[2], p3[2], p4[3];
    float mid[3];

    k2 = k3 = k4 = 0;
    for(i=0; i<3; i++)
    {
        if(connect[im2][i] != im1) p2[k2++] = connect[im2][i];
        if(connect[im3][i] != im1) p3[k3++] = connect[im3][i];
        if(connect[im4][i] != im1) p4[k4++] = connect[im4][i];
    }

    interpolatePoint(im2, p2[0], mid);
    add_res(mid,0);

    if(p3[0]==p2[0] || p3[1] == p2[0])
    {
        if(p3[0] == p2[0])
        {
            interpolatePoint(im3, p3[0], mid);
            add_res(mid,1);
            interpolatePoint(im3, p3[1], mid);
            add_res(mid,2);
        }
        else
        {
            interpolatePoint(im3, p3[1], mid);
            add_res(mid,1);
        }
    }
}

```

```

        interpolatePoint(im3, p3[0], mid);
        add_res(mid,2);
    }

    if(p4[0] == p2[1])
    {
        interpolatePoint(im4, p4[1], mid);
        add_res(mid,3);
        interpolatePoint(im4, p4[0], mid);
        add_res(mid,4);
    }
    else
    {
        interpolatePoint(im4, p4[0], mid);
        add_res(mid,3);
        interpolatePoint(im4, p4[1], mid);
        add_res(mid,4);
    }
}
else
{
    if(p4[0] == p2[0])
    {
        interpolatePoint(im4, p4[0], mid);
        add_res(mid,1);
        interpolatePoint(im4, p4[1], mid);
        add_res(mid,2);
    }
    else
    {
        interpolatePoint(im4, p4[1], mid);
        add_res(mid,1);
        interpolatePoint(im4, p4[0], mid);
        add_res(mid,2);
    }
}

    if(p3[0] == p2[1])
    {
        interpolatePoint(im3, p3[1], mid);
        add_res(mid,3);
        interpolatePoint(im3, p3[0], mid);
        add_res(mid,4);
    }
    else
    {
        interpolatePoint(im3, p3[0], mid);
        add_res(mid,3);
        interpolatePoint(im3, p3[1], mid);
        add_res(mid,4);
    }
}

    interpolatePoint(im2, p2[1], mid);
    add_res(mid,5);

    add_to_scene(430);
}

void BbFlow3D::marchingCube431(int im1, int im2, int im3, int im4)
{
    int i, pp[4], p1[2], p4[2], p2, p3;
    int k1, k4;

    float mid[3];

    pp[0] = im1;

```

```

else
{
    interpolatePoint(pp[3], p4[0], mid);
    add_res(mid,2);
    interpolatePoint(pp[3], p4[1], mid);
    add_res(mid,3);
}
interpolatePoint(pp[1], p2, mid);
add_res(mid,4);
interpolatePoint(pp[0], p1[0], mid);
add_res(mid,5);
}

add_to_scene(431);
}

void BbFlow3D::add_res(float *mid, int k)
{
    for(int j=0; j<3; j++)
        res[k][j] = mid[j];
}

void BbFlow3D::set_scene()
{
    SoSeparator *root = (SoSeparator *) (_objMag->_root);
    SoSeparator *obj = (SoSeparator *) (root->getChild(2));
    SoSeparator *vessels = (SoSeparator *) (obj->getChild(0));

    SoSeparator *s = new SoSeparator;
    vessels -> addChild(s);

    SoCoordinate3 *pCoord = new SoCoordinate3;
    s -> addChild(pCoord);

    SoIndexedFaceSet *pFace = new SoIndexedFaceSet;
    s -> addChild(pFace);

    pCoord->point.setValues(0, _pts, _vertex);
    pFace->coordIndex.setValues(0, _indexNum, _index);
}

void BbFlow3D::add_to_scene(int k)
{
    int i, n, m, tmp;
    n = m = 0;

    if(_pts < (MAXPTS - 10) && _indexNum < (MAXINDEX-20))
    {
        switch (k)
        {
            case 1:
                n = 3;
                m = 4;
                add_coord(n);
                for(i=0; i<m; i++)
                {
                    if(_FaceIndex1[i] == -1) tmp = 0;
                    else tmp = _pts;
                    _index[_indexNum + i] = _FaceIndex1[i] + tmp;
                }
                break;
            case 2:
                n = 4;
                m = 8;
                add_coord(n);

```

```

    for(i=0; i<m; i++)
    {
        if(_FaceIndex2[i] == -1) tmp = 0;
        else tmp = _pts;
        _index[_indexNum + i] = _FaceIndex2[i] + tmp;
    }
    break;
case 3:
    n = 5;
    m = 12;
    add_coord(n);
    for(i=0; i<m; i++)
    {
        if(_FaceIndex3[i] == -1) tmp = 0;
        else tmp = _pts;
        _index[_indexNum + i] = _FaceIndex3[i] + tmp;
    }
    break;
case 44:
    n = 4;
    m = 8;
    add_coord(n);
    for(i=0; i<m; i++)
    {
        if(_FaceIndex2[i] == -1) tmp = 0;
        else tmp = _pts;
        _index[_indexNum + i] = _FaceIndex2[i] + tmp;
    }
    break;
case 430:
    n = 6;
    m = 16;
    add_coord(n);
    for(i=0; i<m; i++)
    {
        if(_FaceIndex430[i] == -1) tmp = 0;
        else tmp = _pts;
        _index[_indexNum + i] = _FaceIndex430[i] + tmp;
    }
    break;
case 431:
    n = 6;
    m = 16;
    add_coord(n);
    for(i=0; i<m; i++)
    {
        if(_FaceIndex431[i] == -1) tmp = 0;
        else tmp = _pts;
        _index[_indexNum + i] = _FaceIndex431[i] + tmp;
    }
    break;
default:
    break;
}
_pts += n;
_indexNum += m;
}
else
{
    printf(" Need to increase number MAXPTS or MAXINDEX \n");
    printf(" _pts = %d _indexNum = %d\n", _pts, _indexNum);
}
}

void BbFlow3D::add_coord(int n)
{

```

```

        *im1 = flag[0];
        *im2 = flag[2];
        *im3 = flag[1];
    }
    else
    {
        *im1 = flag[1];
        *im2 = flag[2];
        *im3 = flag[0];
    }
}
else if(tag == 2)
{
    if(tmp == 11)
    {
        *im1 = flag[0];
        *im2 = flag[1];
        *im3 = flag[2];
    }
    else if(tmp == 101)
    {
        *im1 = flag[1];
        *im2 = flag[0];
        *im3 = flag[2];
    }
    else if(tmp == 110)
    {
        *im1 = flag[2];
        *im2 = flag[0];
        *im3 = flag[1];
    }
}
}

return tag;
}

int BbFlow3D::neighbor4(int *im1, int *im2, int *im3, int *im4)
{
    unsigned char k = 0;
    unsigned char tag = 0;
    unsigned char tmp[4] = {0, 0, 0, 0};
    unsigned char tmp2[8];
    unsigned char i, k2;
    int kc;

    if(neighbor2(flag[0], flag[1]))
    {
        tmp2[k++] = 0;
        tmp2[k++] = 1;
        ++tmp[0];
        ++tmp[1];
        ++tag;
    }
    if(neighbor2(flag[0], flag[2]))
    {
        tmp2[k++] = 0;
        tmp2[k++] = 2;
        ++tmp[0];
        ++tmp[2];
        ++tag;
    }
    if(neighbor2(flag[0], flag[3]))
    {
        tmp2[k++] = 0;
        tmp2[k++] = 3;
        ++tmp[0];
    }
}

```

```

        *im1 = flag[tmp2[0]];
        *im2 = flag[tmp2[1]];
        *im3 = flag[tmp2[2]];
        *im4 = flag[tmp2[3]];
    }
    else
    {
        tag = 31;
        k = 0;
        k2 = 2;
        for(i=0; i<4; i++)
            if(tmp[i] == 2) tmp2[k2++] = i;
            else if(tmp[i] == 1) tmp2[k++] = i;

        *im1 = flag[tmp2[0]];
        *im2 = flag[tmp2[1]];
        *im3 = flag[tmp2[2]];
        *im4 = flag[tmp2[3]];
    }
}

return tag;
}

Boolean BbFlow3D::interPoint(int x, int y, int z)
{
    if(!borderPoint(x, y, z))
    {
        if(threshold(_data3d[y][x-1][z]) && threshold(_data3d[y][x+1][z])
        && threshold(_data3d[y-1][x][z]) && threshold(_data3d[y+1][x][z])
        && threshold(_data3d[y][x][z-1]) && threshold(_data3d[y][x][z+1]))
            return TRUE;
        else return FALSE;
    }
    else return FALSE;
}

Boolean BbFlow3D::borderPoint(int x, int y, int z)
{
    if(x == 0 || x == (_width - 1) ||
       y == 0 || y == (_height - 1) ||
       z == 0 || z == (_depth - 1))
        return TRUE;
    else return FALSE;
}

Boolean BbFlow3D::threshold(short data)
{
    if(float(data) >= _lowThreshold && float(data) <= _highThreshold)
        return TRUE;
    else return FALSE;
}

/*
void BbFlow3D::medial_axis()
{
    int x, y, z;

    for(x=0; x<_xsize; x++)
        for(y=0; y<_ysize; y++)
            for(z=0; z<_zsize; z++)
                _status3d[x][y][z] = 0;

    _numAxis = 0;
    for(i=0; i<_pts; i+)

```

```

    if(_status3d[_vertex[i][0]][_vertex[i][1]][_vertex[i][2]] == 0)
    {
        if(locMax(_vertex[i][0], _vertex[i][1], _vertex[i][2]))
            medial_axis0(_vertex[i][0], _vertex[i][1], _vertex[i][2]);
    }
}

void BbFlow3D::medial_axis0(int x, int y, int z)
{
    int j, num, flag;

    _axisTmp[0][0] = x;
    _axisTmp[0][1] = y;
    _axisTmp[0][2] = z;

    num = 1;

    flag = medial_axis1(x, y, z);

    while(flag == 1)
    {
        _axisTmp[num++][0] = _axisBranch[0][0];
        _axisTmp[num++][1] = _axisBranch[0][1];
        _axisTmp[num++][2] = _axisBranch[0][2];
        flag = medial_axis1(_axisBranch[0][0], _axisBranch[0][1], _axisBranch[0][2]);
    }

    if(num > 1)
    {
        for(j=0; j<num; j++)
        {
            _axis[_numAxis++][j][0] = _axisTmp[j][0];
            _axis[_numAxis++][j][1] = _axisTmp[j][1];
            _axis[_numAxis++][j][2] = _axisTmp[j][2];
        }
    }

    if(flag > 1)
    {
        for(j=0; j<flag; j++)
            medial_axis0(_axisBranch[j][0], _axisBranch[j][1], _axisBranch[j][2]);
    }
}

//
// find all the local maxmum points for i's 26 neighbors
//
int BbFlow3D::medial_axis1(int x, int y, int z)
{
    int num;

    num = 0;
    for(int i=0; i<26; i++)
    {
        get_26_neighbor(i, x, y, z, &x2, &y2, &z2);
        if(_status[x2][y2][z2] == 0 && locMax(x2, y2, z2))
        {
            _axisBranch[num++][0] = x2;
            _axisBranch[num++][1] = y2;
            _axisBranch[num++][2] = z2;
        }
    }
    return num;
}

Boolean BbFlow3D::locMax(int x, int y, int z)

```

```

{
    int x2, y2, z2;

    _status3d[x][y][z] = 1;
    for(int i=0; i<26; i++)
    {
        get_26_neighbor(i, x, y, z, &x2, &y2, &z2);
        if(verify3d(x2, y2, z2) && _data3d[x][y][z] < _data3d[x2][y2][z2]) return FALSE;
    }
    return TRUE;
}

Boolean BbFlow3D::verify3d(int x, int y, int z)
{
    if(x<0 || y<0 || z<0 || x >= _xsize || y>=_ysize || z>=_zsize)
        return FALSE;
    else
        return TRUE;
}

void BbFlow3D::get_26_neighbor(int i, int x, int y, int z,
int *x2, int *y2, int *z2)
{
    switch (i)
    {
        case 0: *x2 = x - 1, *y2 = y - 1, *z2 = z - 1, break;
        case 1: *x2 = x - 1, *y2 = y - 1, *z2 = z, break;
        case 2: *x2 = x - 1, *y2 = y - 1, *z2 = z + 1, break;
        case 3: *x2 = x - 1, *y2 = y, *z2 = z - 1, break;
        case 4: *x2 = x - 1, *y2 = y, *z2 = z, break;
        case 5: *x2 = x - 1, *y2 = y, *z2 = z + 1, break;
        case 6: *x2 = x - 1, *y2 = y + 1, *z2 = z - 1, break;
        case 7: *x2 = x - 1, *y2 = y + 1, *z2 = z, break;
        case 8: *x2 = x - 1, *y2 = y + 1, *z2 = z + 1, break;
        case 9: *x2 = x, *y2 = y - 1, *z2 = z - 1, break;
        case 10: *x2 = x, *y2 = y - 1, *z2 = z, break;
        case 11: *x2 = x, *y2 = y - 1, *z2 = z + 1, break;
        case 12: *x2 = x, *y2 = y, *z2 = z - 1, break;
        case 13: *x2 = x, *y2 = y, *z2 = z + 1, break;
        case 14: *x2 = x, *y2 = y + 1, *z2 = z - 1, break;
        case 15: *x2 = x, *y2 = y + 1, *z2 = z, break;
        case 16: *x2 = x, *y2 = y + 1, *z2 = z + 1, break;
        case 17: *x2 = x + 1, *y2 = y - 1, *z2 = z - 1, break;
        case 18: *x2 = x + 1, *y2 = y - 1, *z2 = z, break;
        case 19: *x2 = x + 1, *y2 = y - 1, *z2 = z + 1, break;
        case 20: *x2 = x + 1, *y2 = y, *z2 = z - 1, break;
        case 21: *x2 = x + 1, *y2 = y, *z2 = z, break;
        case 22: *x2 = x + 1, *y2 = y, *z2 = z + 1, break;
        case 23: *x2 = x + 1, *y2 = y + 1, *z2 = z - 1, break;
        case 24: *x2 = x + 1, *y2 = y + 1, *z2 = z, break;
        case 25: *x2 = x + 1, *y2 = y + 1, *z2 = z + 1, break;
        default: break;
    }
}

*/
//---- End editable code block: End of generated code

```



```

////////////////////////////////////
//
// Source file for BbFlow3DUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbFlow3DUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbFlow3DUI::_defaultBbFlow3DUIResources[] = {
    "*buttonAccept1.labelString: Accept",
    "*buttonSaveFlow3D.labelString: Save 3D",
    "*labelFlow3DEnd.labelString: End",
    "*labelFlow3DHigh.labelString: High",
    "*labelFlow3DLow.labelString: Low",
    "*labelFlow3DStart.labelString: Start",
    "*optionAddCut.labelString: Add Cut",
    "*optionAddObj.labelString: Add Obj",
    "*optionPoint.labelString: Point",
    "*optionSurface.labelString: Surface",
    "*tabLabel: Vessel3D",
    "*toggleDisable.labelString: Disable",
    "*toggleEnable.labelString: Enable",

    //---- Start editable code block: BbFlow3DUI Default Resources

    //---- End editable code block: BbFlow3DUI Default Resources

```

};

BbFlow3DUI::BbFlow3DUI (const char *name) : VkComponent (name)

```
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbFlow3D constructor 2

    //---- End editable code block: BbFlow3D constructor 2

} // End Constructor
```

BbFlow3DUI::BbFlow3DUI (const char *name, Widget parent) : VkComponent (name)

```
{
    //---- Start editable code block: BbFlow3D pre-create

    //---- End editable code block: BbFlow3D pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: BbFlow3D constructor

    //---- End editable code block: BbFlow3D constructor

} // End Constructor
```

BbFlow3DUI::~BbFlow3DUI()

```
{
    // Base class destroys widgets

    //---- Start editable code block: BbFlow3DUI destructor

    //---- End editable code block: BbFlow3DUI destructor
} // End destructor
```

void BbFlow3DUI::create (Widget parent)

```
{
    Arg      args[7];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBbFlow3DUIResources );
}
```

[illegible]

```

_textfieldFlow3DStartXtVaCreateManagedWidget ( "textfieldFlow3DStart",
XmTextFieldWidgetClass,
_baseWidget,
XmNcolumns, 7,
XmNx, 200,
XmNy, 10,
XmNheight, 35,
(XtPointer) NULL );

```

[illegible]

```
XtAddCallback ( _buttonSaveFlow3D,  
                XmNactivateCallback,  
                &BbFlow3DUI::doButtonSaveFlow3DCallback,  
                (XtPointer) this );
```

[illegible][illegible][illegible][illegible]

```
_textfield1 = XtVaCreateManagedWidget ( "textfield1",
                                         xmTextFieldWidgetClass,
                                         _baseWidget,
                                         XmNcolumns, 10,
                                         XmNx, 449,
                                         XmNy, 59,
                                         XmNheight, 35,
                                         (XtPointer) NULL );

_buttonAccept1 = XtVaCreateManagedWidget ( "buttonAccept1",
                                             xmPushButtonWidgetClass,
                                             _baseWidget,
                                             XmNlabelType, XmSTRING,
                                             XmNx, 450,
                                             XmNy, 10,
                                             XmNwidth, 100,
                                             XmNheight, 40,
                                             (XtPointer) NULL );

XtAddCallback ( _buttonAccept1,
                XmNactivateCallback,
                &BbFlow3DUI::doButtonAcceptCallback,
                (XtPointer) this );

_radiobox = XtVaCreateManagedWidget ( "radiobox",
                                       xmRowColumnWidgetClass,
                                       _baseWidget,
                                       XmNpacking, XmPACK_COLUMN,
                                       XmNradioBehavior, True,
                                       XmNradioAlwaysOne, True,
                                       XmNx, 30,
                                       XmNy, 27,
                                       XmNwidth, 79,
                                       XmNheight, 61,
                                       (XtPointer) NULL );

_toggleDisable = XtVaCreateManagedWidget ( "toggleDisable",
                                             xmToggleButtonWidgetClass,
                                             _radiobox,
                                             XmNlabelType, XmSTRING,
                                             (XtPointer) NULL );

XtAddCallback ( _toggleDisable,
                XmNvalueChangedCallback,
                &BbFlow3DUI::setToggleDisableCallback,
                (XtPointer) this );

_toggleEnable = XtVaCreateManagedWidget ( "toggleEnable",
                                            xmToggleButtonWidgetClass,
                                            _radiobox,
                                            XmNlabelType, XmSTRING,
                                            (XtPointer) NULL );

XtAddCallback ( _toggleEnable,
                XmNvalueChangedCallback,
                &BbFlow3DUI::setToggleEnableCallback,
                (XtPointer) this );
```

```

XtVaSetValues ( _optionMenu12->baseWidget(),
                XmNx, 110,
                XmNy, 110,
                XmNwidth, 115,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenu10->baseWidget(),
                XmNx, 170,
                XmNy, 109,
                XmNwidth, 119,
                XmNheight, 32,
                (XtPointer) NULL );

//---- Start editable code block: BbFlow3DUI create
XmToggleButtonSetState(_toggleDisable,TRUE,TRUE);

//---- End editable code block: BbFlow3DUI create
}

const char * BbFlow3DUI::className()
{
    return ("BbFlow3DUI");
} // End className()

////////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////////

void BbFlow3DUI::doButtonAcceptCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doButtonAccept ( w, callData );
}

void BbFlow3DUI::doButtonSaveFlow3DCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doButtonSaveFlow3D ( w, callData );
}

void BbFlow3DUI::doOptionAddCutCallback ( Widget      w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doOptionAddCut ( w, callData );
}

void BbFlow3DUI::doOptionAddObjCallback ( Widget      w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doOptionAddObj ( w, callData );
}

void BbFlow3DUI::doOptionPointCallback ( Widget      w,

```

```
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doOptionPoint ( w, callData );
}

void BbFlow3DUI::doOptionSurfaceCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->doOptionSurface ( w, callData );
}

void BbFlow3DUI::setToggleDisableCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->setToggleDisable ( w, callData );
}

void BbFlow3DUI::setToggleEnableCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbFlow3DUI* obj = ( BbFlow3DUI * ) clientData;
    obj->setToggleEnable ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbFlow3DUI::doButtonAccept ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::doButtonSaveFlow3D ( Widget, XtPointer )
{
    // This virtual function is called from doButtonSaveFlow3DCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::doOptionAddCut ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAddCutCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::doOptionAddObj ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAddObjCallback.
    // This function is normally overridden by a derived class.
}
```

```
void BbFlow3DUI::doOptionPoint ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPointCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::doOptionSurface ( Widget, XtPointer )
{
    // This virtual function is called from doOptionSurfaceCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::setToggleDisable ( Widget, XtPointer )
{
    // This virtual function is called from setToggleDisableCallback.
    // This function is normally overridden by a derived class.
}

void BbFlow3DUI::setToggleEnable ( Widget, XtPointer )
{
    // This virtual function is called from setToggleEnableCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```


User: meide
Host: phoenix
Class: phoenix
Job: BbDisplay.C

316

```

////////////////////////////////////
//
// Source file for BbFlowUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbFlowUI.h" // Generated header file for this class

#include <Sgm/ThumbWheel.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbFlowUI::_defaultBbFlowUIResources[] = {
    "buttonAcceptFlow.labelString: Accept ",
    "labelArea.labelString: Area",
    "labelBSV.labelString: BSV",
    "labelMV.labelString: MV",
    "labelPSV.labelString: PSV",
    "labelVFR.labelString: VFR",
    "optionAutoEdge.labelString: Auto Edge",
    "optionAutoSnake.labelString: Auto Snake",
    "optionAutoThresh.labelString: Auto Thresh",
    "optionManual.labelString: Manual",
    "optionMenuFlowMethod.labelString: ",
    "optionSemiAuto.labelString: Semi Auto",
    "tabLabel: Flow",

    //---- Start editable code block: BbFlowUI Default Resources

    //---- End editable code block: BbFlowUI Default Resources

    (char*)NULL

```

```
BbFlowUI::BbFlowUI ( const char *name ) : VkComponent ( name )
```

```
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbFlow constructor 2

    //---- End editable code block: BbFlow constructor 2

}    // End Constructor
```

```
BbFlowUI::BbFlowUI ( const char *name, Widget parent ) : VkComponent ( name )
```

```
{
    //---- Start editable code block: BbFlow pre-create

    //---- End editable code block: BbFlow pre-create

    // Call creation function to build the widget tree.
    .create ( parent );

    //---- Start editable code block: BbFlow constructor

    //---- End editable code block: BbFlow constructor

}    // End Constructor
```

```
BbFlowUI::~BbFlowUI()
```

```
{
    // Base class destroys widgets

    //---- Start editable code block: BbFlowUI destructor

    //---- End editable code block: BbFlowUI destructor
}    // End destructor
```

```
void BbFlowUI::create ( Widget parent )
```

```
{
    Arg      args[11];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object

    setDefaultResources ( parent, _defaultBbFlowUIResources );
```

```
// Create an unmanaged widget as the top of the widget hierarchy
```

```
_baseWidget = _bbFlow = XtVaCreateWidget ( _name,
                                             xmBulletinBoardWidgetClass,
                                             parent,
                                             XmNresizePolicy, XmRESIZE_GROW,
                                             (XtPointer) NULL );
```

```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component
```

```
// All variables are data members of this class
```

```
_textfieldMin = XtVaCreateManagedWidget ( "textfieldMin",
                                             xmTextFieldWidgetClass,
                                             _baseWidget,
                                             XmNcolumns, 5,
                                             XmNx, 420,
                                             XmNy, 60,
                                             XmNheight, 35,
                                             (XtPointer) NULL );
```

```
XtAddCallback ( _textfieldMin,
                 XmNactivateCallback,
                 &BbFlowUI::minFlowCallback,
                 (XtPointer) this );
```

```
_textfieldMax = XtVaCreateManagedWidget ( "textfieldMax",
                                             xmTextFieldWidgetClass,
                                             _baseWidget,
                                             XmNcolumns, 5,
                                             XmNx, 508,
                                             XmNy, 60,
                                             XmNheight, 35,
                                             (XtPointer) NULL );
```

```
XtAddCallback ( _textfieldMax,
                 XmNactivateCallback,
                 &BbFlowUI::maxFlowCallback,
                 (XtPointer) this );
```

```
_thumbwheelSemi = XtVaCreateManagedWidget ( "thumbwheelSemi",
                                             sgThumbWheelWidgetClass,
                                             _baseWidget,
                                             XmNmaximum, 720,
                                             SgNhomePosition, 0,
                                             SgNangleRange, 720,
                                             SgNunitsPerRotation, 1,
                                             XmNvalue, 0,
                                             XmNOrientation, XmHORIZONTAL,
                                             XmNx, 420,
                                             XmNy, 109,
                                             XmNwidth, 152,
                                             XmNheight, 24,
                                             (XtPointer) NULL );
```

```
XtAddCallback ( _thumbwheelSemi,
                 XmNvalueChangedCallback,
                 &BbFlowUI::SemiFlowChgCallback,
                 (XtPointer) this );
```

```

XtAddCallback ( _thumbWheelSemi,
                XmNdragCallback,
                &BbFlowUI::SemiFlowCallback,
                (XtPointer) this );

```

320

```

_optionMenuFlowMethod = new VkOptionMenu ( _baseWidget, "optionMenuFlowMethod");
_optionManual = _optionMenuFlowMethod->addAction ( "optionManual",
                                                    &BbFlowUI::doOptionManualCallba
                                                    (XtPointer) this );

_optionSemiAuto = _optionMenuFlowMethod->addAction ( "optionSemiAuto",
                                                    &BbFlowUI::doOptionSemiAutoCa
                                                    (XtPointer) this );

_optionAutoSnake = _optionMenuFlowMethod->addAction ( "optionAutoSnake",
                                                    &BbFlowUI::doOptionAutoSnake
                                                    (XtPointer) this );

_optionAutoEdge = _optionMenuFlowMethod->addAction ( "optionAutoEdge",
                                                    &BbFlowUI::doOptionAutoEdgeCa
                                                    (XtPointer) this );

_optionAutoThresh = _optionMenuFlowMethod->addAction ( "optionAutoThresh",
                                                    &BbFlowUI::doOptionAutoThre
                                                    (XtPointer) this );

_labelArea = XtVaCreateManagedWidget ( "labelArea",
                                       xmLabelWidgetClass,
                                       _baseWidget,
                                       XmNlabelType, XmSTRING,
                                       XmNx, 255,
                                       XmNy, 85,
                                       XmNwidth, 36,
                                       XmNheight, 20,
                                       (XtPointer) NULL );

_labelMV = XtVaCreateManagedWidget ( "labelMV",
                                       xmLabelWidgetClass,
                                       _baseWidget,
                                       XmNlabelType, XmSTRING,
                                       XmNx, 255,
                                       XmNy, 33,
                                       XmNwidth, 31,
                                       XmNheight, 30,
                                       (XtPointer) NULL );

_labelBSV = XtVaCreateManagedWidget ( "labelBSV",
                                       xmLabelWidgetClass,
                                       _baseWidget,
                                       XmNlabelType, XmSTRING,
                                       XmNx, 118,
                                       XmNy, 117,
                                       XmNwidth, 34,
                                       XmNheight, 20,
                                       (XtPointer) NULL );

_labelPSV = XtVaCreateManagedWidget ( "labelPSV",
                                       xmLabelWidgetClass,
                                       _baseWidget,
                                       XmNlabelType, XmSTRING,
                                       XmNx, 117,
                                       XmNy, 71,

```


XmNx, 155,
XmNy, 63,
XmNheight, 35,
(XtPointer) NULL);

322

```
_textfieldVFR = XtVaCreateManagedWidget ( "textfieldVFR",  
                                           xmTextFieldWidgetClass,  
                                           _baseWidget,  
                                           XmNcolumns, 7,  
                                           XmNx, 155,  
                                           XmNy, 16,  
                                           XmNheight, 35,  
                                           (XtPointer) NULL );
```

```
XtVaSetValues ( _optionMenuFlowMethod->baseWidget(),  
               XmNx, 420,  
               XmNy, 10,  
               XmNwidth, 145,  
               XmNheight, 32,  
               (XtPointer) NULL );
```

```
//---- Start editable code block: BbFlowUI create
```

```
//---- End editable code block: BbFlowUI create
```

```
}
```

```
const char * BbFlowUI::className()  
{  
    return ("BbFlowUI");  
} // End className()
```

```
////////////////////////////////////  
// The following functions are static member functions used to  
// interface with Motif.  
////////////////////////////////////
```

```
void BbFlowUI::SemiFlowCallback ( Widget    w,  
                                 XtPointer clientData,  
                                 XtPointer callData )
```

```
{  
    BbFlowUI* obj = ( BbFlowUI * ) clientData;  
    obj->SemiFlow ( w, callData );  
}
```

```
void BbFlowUI::SemiFlowChgCallback ( Widget    w,  
                                    XtPointer clientData,  
                                    XtPointer callData )
```

```
{  
    BbFlowUI* obj = ( BbFlowUI * ) clientData;  
    obj->SemiFlowChg ( w, callData );  
}
```

```
void BbFlowUI::doButtonAcceptFlowCallback ( Widget    w,  
                                             XtPointer clientData,  
                                             XtPointer callData )
```

```
{  
    BbFlowUI* obj = ( BbFlowUI * ) clientData;  
    obj->doButtonAcceptFlow ( w, callData );  
}
```

```

void BbFlowUI::doOptionAutoEdgeCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->doOptionAutoEdge ( w, callData );
}

void BbFlowUI::doOptionAutoSnakeCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->doOptionAutoSnake ( w, callData );
}

void BbFlowUI::doOptionAutoThreshCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->doOptionAutoThresh ( w, callData );
}

void BbFlowUI::doOptionManualCallback ( Widget      w,
                                        XtPointer clientData,
                                        XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->doOptionManual ( w, callData );
}

void BbFlowUI::doOptionSemiAutoCallback ( Widget      w,
                                        XtPointer clientData,
                                        XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->doOptionSemiAuto ( w, callData );
}

void BbFlowUI::maxFlowCallback ( Widget      w,
                                XtPointer clientData,
                                XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->maxFlow ( w, callData );
}

void BbFlowUI::minFlowCallback ( Widget      w,
                                XtPointer clientData,
                                XtPointer callData )

{
    BbFlowUI* obj = ( BbFlowUI * ) clientData;
    obj->minFlow ( w, callData );
}

```

```

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

```

```

void BbFlowUI::SemiFlow ( Widget, XtPointer )
{
    // This virtual function is called from SemiFlowCallback.
}

```



```
// This function is normally overridden by a derived class.
```

324

```
}
```

```
void BbFlowUI::SemiFlowChg ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from SemiFlowChgCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doButtonAcceptFlow ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doButtonAcceptFlowCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doOptionAutoEdge ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doOptionAutoEdgeCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doOptionAutoSnake ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doOptionAutoSnakeCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doOptionAutoThresh ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doOptionAutoThreshCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doOptionManual ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doOptionManualCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::doOptionSemiAuto ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from doOptionSemiAutoCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::maxFlow ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from maxFlowCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbFlowUI::minFlow ( Widget, XtPointer )
```

```
{  
    // This virtual function is called from minFlowCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbFormat
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFormatUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbFormat.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbFormatUI and are
// available as protected data members inherited by this class
//
// XmPushButton                _buttonPublishPath
// XmTextField                 _textfieldNewPath1
// VkOptionMenu *              _optionMenu16
// VkMenuItem *                _optionPublishNone
// VkMenuItem *                _option2DMag
// VkMenuItem *                _option2DPhase
// VkMenuItem *                _option2DWave
// VkMenuItem *                _option2DLoc
// VkMenuItem *                _option3DLoc
// VkMenuItem *                _option3DFlow
// VkMenuItem *                _optionHTML
// VkMenuItem *                _optionMPEG
// VkMenuItem *                _optionPublishArea
// VkMenuItem *                _optionPublishShear
// XmPushButton                _button3DContour
// XmPushButton                _buttonShow2DContour
// VkOptionMenu *              _optionMenu13
// VkMenuItem *                _optionWholeImg
// VkMenuItem *                _optionROI3
// VkMenuItem *                _option3DLocLarge
// VkMenuItem *                _option3DLocSmall
// VkMenuItem *                _option3DFlowLarge
// VkMenuItem *                _option3DFlowSmall
// VkOptionMenu *              _optionMenu9
// VkMenuItem *                _optionGIF

```

```

// VkMenuItem *           _optionTIFF
// VkMenuItem *           _optionRGB
// XmTextField             _textfieldExtension
// XmTextField             _textfieldFname
// XmPushButton            _buttonAcceptFlow1
// XmLabel                 _labelFname
// XmLabel                 _labelFnameExt
//

```

```

////////////////////////////////////

```

```

//---- Start editable code block: headers and declarations

```

```

#include <Inventor/nodes/SoCoordinate3.h>
#include <Inventor/nodes/SoPointSet.h>
#include <Inventor/nodes/SoDrawStyle.h>
#include <Inventor/nodes/SoMaterial.h>
#include <Inventor/Xt/viewers/SoXtExaminerViewer.h>
#include "Bb.h"
#include "Utility.h"
#include <Vk/VkFileSelectionDialog.h>
#include "BbVisual.h"

```

```

//---- End editable code block: headers and declarations

```

```

//---- BbFormat Constructor

```

```

BbFormat::BbFormat(const char *name, Widget parent) :
    BbFormatUI(name, parent)
{
    // This constructor calls BbFormatUI(parent, name)
    // which calls BbFormatUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbFormat constructor

    init();

    //---- End editable code block: BbFormat constructor

}    // End Constructor

```

```

BbFormat::BbFormat(const char *name) :
    BbFormatUI(name)
{
    // This constructor calls BbFormatUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbFormat constructor 2

    init();

    //---- End editable code block: BbFormat constructor 2

}    // End Constructor

```

```

BbFormat::~BbFormat()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //----- Start editable code block: BbFormat destructor

    //----- End editable code block: BbFormat destructor

}    // End Destructor

const char * BbFormat::className() // classname
{
    return ("BbFormat");
} // End className()

void BbFormat::doButton3DContour ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doButton3DContour

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doButton3DContour is implemented
    //::VkUnimplemented ( w, "BbFormat::doButton3DContour" );

    float  r, g, b;

    if(_objMag->_imgView2->_roi_color == COLOR_GREEN)
    {
        r = 0, g=1, b=0;
    }
    else if(_objMag->_imgView2->_roi_color == COLOR_BLUE)
    {
        r = 0, g=0, b=1;
    }
    else if(_objMag->_imgView2->_roi_color == COLOR_WHITE)
    {
        r = 1, g=1, b=1;
    }
    else if(_objMag->_imgView2->_roi_color == COLOR_YELLOW)
    {
        r = 1, g=1, b=0;
    }
    else
    {
        r = 1, g=1, b=1;
    }

    float thresh;
    FILE *fp = fopen("out.con","r");
    int tmp;
    fscanf(fp, "%d %d", &tmp, &tmp);
    Points *p = new Points();
    p -> from_ContourFile(fp);
    fscanf(fp, "%f", &thresh);
    fclose(fp);
}

```

```

int    num;
float vertex[1000][3];
float x, y;

float zoom;
zoom = _objMag -> _imgView2 -> _zoom;
short **img = _objMag -> _img2 -> get_imgdata();
//zoom = 1.0;
//short **img = _objMag -> _imgView2 -> _zoomImg;

num = p -> _numPoints;

int i;

for(i=0; i<num; i++)
{
    x = p -> _points[i].x / zoom;
    y = p -> _points[i].y / zoom;
    vertex[i][0] = x;
    vertex[i][1] = float(img[int(y)][int(x)]); // * (_objMag -> msgsRight.ratio3D)
    vertex[i][2] = y;
}

float val, x2, y2;

if(thresh != 1.0)
for(i=0; i<num; i++)
{
    val = fabs(vertex[i][1]);
    if(val > thresh)
    {
        x = vertex[i][0];
        y = vertex[i][2]-1;
        if(val > fabs(img[int(y)][int(x)]))
        {
            x2 = x;
            y2 = y;
            val = fabs(img[int(y)][int(x)]);
        }

        x = vertex[i][0];
        y = vertex[i][2]+1;
        if(val > fabs(img[int(y)][int(x)]))
        {
            x2 = x;
            y2 = y;
            val = fabs(img[int(y)][int(x)]);
        }

        x = vertex[i][0]-1;
        y = vertex[i][2];
        if(val > fabs(img[int(y)][int(x)]))
        {
            x2 = x;
            y2 = y;
            val = fabs(img[int(y)][int(x)]);
        }

        x = vertex[i][0]+1;
        y = vertex[i][2];
        if(val > fabs(img[int(y)][int(x)]))
        {
            x2 = x;
            y2 = y;
            val = fabs(img[int(y)][int(x)]);
        }
    }
}

```

```

x = vertex[i][0]-1;
y = vertex[i][2]-1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]+1;
y = vertex[i][2]-1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]-1;
y = vertex[i][2]+1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]+1;
y = vertex[i][2]+1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

vertex[i][0] = x2;
vertex[i][2] = y2;
vertex[i][1] = img[int(y2)][int(x2)];

/*
if(val > thresh)
{
    x = vertex[i][0];
    y = vertex[i][2]-2;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0];
    y = vertex[i][2]+2;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0]-2;
    y = vertex[i][2];
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
}

```

```

}
x = vertex[i][0]+2;
y = vertex[i][2];
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]-2;
y = vertex[i][2]-2;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]-1;
y = vertex[i][2]-2;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]+1;
y = vertex[i][2]-2;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]+2;
y = vertex[i][2]-2;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]-2;
y = vertex[i][2]-1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]+2;
y = vertex[i][2]-1;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

x = vertex[i][0]-2;
y = vertex[i][2]+2;
if(val > fabs(img[int(y)][int(x)]))
{
    x2 = x;
    y2 = y;
    val = fabs(img[int(y)][int(x)]);
}

```



```

    }
    x = vertex[i][0]-1;
    y = vertex[i][2]+2;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0]+1;
    y = vertex[i][2]+2;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0]+2;
    y = vertex[i][2]+2;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0]-2;
    y = vertex[i][2]+1;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    x = vertex[i][0]+2;
    y = vertex[i][2]+1;
    if(val > fabs(img[int(y)][int(x)]))
    {
        x2 = x;
        y2 = y;
        val = fabs(img[int(y)][int(x)]);
    }
    vertex[i][0] = x2;
    vertex[i][2] = y2;
    vertex[i][1] = img[int(y2)][int(x2)];
}
*/
}

int j, kc, cur;
float vertex2[500][3];

cur = 1;
for(j=0; j<3; j++)
    vertex2[0][j] = vertex[0][j];

for(i=0; i<num; i++)
{
    kc = 0;
    for(j=0; j<cur; j++)
        if(int(vertex[i][0]) == int(vertex[j][0])
            && int(vertex[i][2]) == int(vertex[j][2]))
            {kc = 1; break;}
    if(kc == 0)
    {

```

```

    for(j=0; j<3; j++)
        vertex2[cur][j] = vertex[i][j];
    ++cur;
}
}

for(i=0; i<cur; i++)
    vertex2[i][1] *= (_objMag -> msgsRight.ratio3D);

p -> clear();
for(i=0; i<cur; i++)
    p -> add(vertex2[i][0]*zoom, vertex2[i][2]*zoom);

fp = fopen("out2.con", "w");
fprintf(fp, "1010\n1\n");
p -> to_ContourFile(fp);
fprintf(fp, "%f\n", thresh);
fclose(fp);

SoSeparator *root = (SoSeparator *) (( SoXtExaminerViewer *)
    (_objMag -> _R3D ))-> getSceneGraph();

SoSeparator *contour = new SoSeparator;

SoMaterial *ballMaterial = new SoMaterial;
contour -> addChild(ballMaterial);

ballMaterial->ambientColor.setValue(r, g, b);
ballMaterial->diffuseColor.setValue(r, g, b);
ballMaterial->emissiveColor.setValue(r, g, b);
//ballMaterial->transparency.setValue(0.5);

SoDrawStyle *velDrawStyle = new SoDrawStyle;
contour -> addChild(velDrawStyle);

SoCoordinate3 *velCoord = new SoCoordinate3;
contour -> addChild(velCoord);

SoPointSet *velPointSet = new SoPointSet;
contour -> addChild(velPointSet);

velDrawStyle -> style.setValue(velDrawStyle->POINTS);
velDrawStyle -> pointSize.setValue(5);
velCoord -> point.setValues(0, cur, vertex2);
velPointSet -> startIndex.setValue(0);
velPointSet -> numPoints.setValue(cur);

root -> addChild(contour);

delete p;

//---- End editable code block: BbFormat doButton3DContour
} // End BbFormat::doButton3DContour()

void BbFormat::doButtonAcceptFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doButtonAcceptFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doButtonAcceptFlow is implement

    //::VkUnimplemented ( w, "BbFormat::doButtonAcceptFlow" );

```

```

int n = _objMag -> Left.img_number;
char fname[300];
char str_n[20];
sprintf(fname, "%s.%d", XmTextFieldGetString(_textfieldFname), n);
sprintf(str_n, "%d", n);
XmTextFieldSetString(_textfieldExtension, str_n);

toFile(fname);

//---- End editable code block: BbFormat doButtonAcceptFlow
} // End BbFormat::doButtonAcceptFlow()

void BbFormat::doButtonPublishPath ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doButtonPublishPath

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doButtonPublishPath is implemer

    //::VkUnimplemented ( w, "BbFormat::doButtonPublishPath" );
    char str[200];

    if(theFileSelectionDialog->postAndWait() == VkDialogManager::OK)
    {
        sprintf(str, "%s", theFileSelectionDialog->fileName());
        XmTextFieldSetString(_textfieldNewPath1, str);
    }

    //---- End editable code block: BbFormat doButtonPublishPath
} // End BbFormat::doButtonPublishPath()

void BbFormat::doButtonShow2DContour ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doButtonShow2DContour

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doButtonShow2DContour is implen

    //::VkUnimplemented ( w, "BbFormat::doButtonShow2DContour" );

    FILE *fp = fopen("out.con", "r");
    int tmp;
    fscanf(fp, "%d %d", &tmp, &tmp);
    Points *p = new Points();
    p -> from_ContourFile(fp);
    //_objMag->_imgView2->_ROI->_points_in_border.get_Points(zoom, 0, 0);
    //p -> to_ContourFile(fp);
    //fprintf(fp, "1.0\n");
    fclose(fp);

    p -> draw(_objMag->_imgView2->baseWidget(),
        _objMag->_imgView2->_roi_color);

    delete p;

    //---- End editable code block: BbFormat doButtonShow2DContour
}

```

```
) // End BbFormat::doOption2DContour()
```

335

```
void BbFormat::doOption2DLoc ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption2DLoc
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption2DLoc is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOption2DLoc" );
    savePublish(PUBLISH_2DLOC);

```

```
    //---- End editable code block: BbFormat doOption2DLoc
} // End BbFormat::doOption2DLoc()
```

```
void BbFormat::doOption2DMag ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption2DMag
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption2DMag is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOption2DMag" );
    savePublish(PUBLISH_2DMAG);

```

```
    //---- End editable code block: BbFormat doOption2DMag
} // End BbFormat::doOption2DMag()
```

```
void BbFormat::doOption2DPhase ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption2DPhase
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption2DPhase is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOption2DPhase" );
    savePublish(PUBLISH_2DPHA);
    //---- End editable code block: BbFormat doOption2DPhase
} // End BbFormat::doOption2DPhase()
```

```
void BbFormat::doOption2DWave ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption2DWave
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption2DWave is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOption2DWave" );
    savePublish(PUBLISH_2DWAVE);

```

```

char str[200];

sprintf(str, "%sindex.html", XmTextFieldGetString(_textfieldNewPath1));
toHTMLFile(str, 1);
sprintf(str, "%sflow3D.html", XmTextFieldGetString(_textfieldNewPath1));
toHTMLFile(str, 2);

//---- End editable code block: BbFormat doOption2DWave
} // End BbFormat::doOption2DWave()

void BbFormat::doOption3DFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption3DFlow
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption3DFlow is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOption3DFlow" );
    savePublish(PUBLISH_3DFLOW);
    //---- End editable code block: BbFormat doOption3DFlow
} // End BbFormat::doOption3DFlow()

void BbFormat::doOption3DFlowLarge ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption3DFlowLarge
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption3DFlowLarge is implemer
    //::VkUnimplemented ( w, "BbFormat::doOption3DFlowLarge" );
    XtVaSetValues ( _objMag -> _R3D->getWidget(),
                    XmNx, 670,
                    XmNy, 90,
                    XmNwidth, 500,
                    XmNheight, 500,
                    (XtPointer) NULL );
    _type = IMAGE_3DFLOWLARGE;

    //---- End editable code block: BbFormat doOption3DFlowLarge
} // End BbFormat::doOption3DFlowLarge()

void BbFormat::doOption3DFlowSmall ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOption3DFlowSmall
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOption3DFlowSmall is implemer
    //::VkUnimplemented ( w, "BbFormat::doOption3DFlowSmall" );
    XtVaSetValues ( _objMag -> _R3D->getWidget(),
                    XmNx, 765,
                    XmNy, 185,
                    XmNwidth, 306,
                    XmNheight, 306,

```

```
        (XtPointer) NULL );  
_type = IMAGE_3DFLOW; ALL;
```

337

```
    //---- End editable code block: BbFormat doOption3DFlowSmall  
}  
    // End BbFormat::doOption3DFlowSmall()  
  
void BbFormat::doOption3DLoc ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbFormat doOption3DLoc  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when BbFormat::doOption3DLoc is implemented:  
    //::VkUnimplemented ( w, "BbFormat::doOption3DLoc" );  
    savePublish(PUBLISH_3DLOC);  
  
    //---- End editable code block: BbFormat doOption3DLoc  
}  
    // End BbFormat::doOption3DLoc()  
  
void BbFormat::doOption3DLocLarge ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbFormat doOption3DLocLarge  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when BbFormat::doOption3DLocLarge is implement  
    //::VkUnimplemented ( w, "BbFormat::doOption3DLocLarge" );  
  
    XResizeWindow(XtDisplay(_objMag ->_win3D->baseWidget()),  
        XtWindow(_objMag ->_win3D->baseWidget()), 600, 700);  
    _type = IMAGE_3DLOCLARGE;  
  
    //---- End editable code block: BbFormat doOption3DLocLarge  
}  
    // End BbFormat::doOption3DLocLarge()  
  
void BbFormat::doOption3DLocSmall ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbFormat doOption3DLocSmall  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when BbFormat::doOption3DLocSmall is implement  
    //::VkUnimplemented ( w, "BbFormat::doOption3DLocSmall" );  
    XResizeWindow(XtDisplay(_objMag ->_win3D->baseWidget()),  
        XtWindow(_objMag ->_win3D->baseWidget()), 256, 256);  
    _type = IMAGE_3DLOCSMALL;  
  
    //---- End editable code block: BbFormat doOption3DLocSmall  
}  
    // End BbFormat::doOption3DLocSmall()
```

```

void BbFormat::doOptionGIF ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionGIF

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionGIF is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionGIF" );

    _format = IMAGE_GIF;

    //----- End editable code block: BbFormat doOptionGIF
}    // End BbFormat::doOptionGIF()

void BbFormat::doOptionHTML ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionHTML

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionHTML is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionHTML" );
    char str[200];

    sprintf(str, "%sindex.html", XmTextFieldGetString(_textfieldNewPath1));
    toHTMLFile(str, 1);
    sprintf(str, "%sflow3D.html", XmTextFieldGetString(_textfieldNewPath1));
    toHTMLFile(str, 2);

    //----- End editable code block: BbFormat doOptionHTML
}    // End BbFormat::doOptionHTML()

void BbFormat::doOptionMPEG ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionMPEG

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionMPEG is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionMPEG" );

    toMPEG();

    //----- End editable code block: BbFormat doOptionMPEG
}    // End BbFormat::doOptionMPEG()

void BbFormat::doOptionPublishArea ( Widget wid, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionPublishArea

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionPublishArea is implemer
    //::VkUnimplemented ( w, "BbFormat::doOptionPublishArea" );

```

```

if(_objMag -> _imgView2 -> _ROI != NULL)
{
    _objMag -> _imgView2 -> AcceptROI();
    unsigned char **area = _objMag -> _imgView2 -> _ROI -> _area;
    int w = _objMag -> _imgView2 -> get_width();
    int h = _objMag -> _imgView2 -> get_height();

    int n = 0;
    for(int i=0; i<h; i++)
    for(int j=0; j<w; j++)
        if(area[i][j] == 1) n++;
    float z = _objMag -> _imgView2 -> _zoom;
    float a = float(n)/(z*z);

    GE_PCMRA_HEADER_OBJ *pc = _objMag -> _img -> get_header();
    float pixel_area = pc->pixsize_X * pc->pixsize_Y / 100.0;
    a *= pixel_area;
    float r = sqrtf(a/3.141592654);
    printf(" Shear::  n = %d zoom = %6.2f pX=%7.3f pY=%7.3f\n", n, z, pc->pixsize_X,

    char str[50];
    sprintf(str, "%6.3f", 2.0*r);
    _radius = r;
    XmTextFieldSetString(_textfieldFname, str);
}

//---- End editable code block: BbFormat doOptionPublishArea
} // End BbFormat::doOptionPublishArea()

void BbFormat::doOptionPublishNone ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOptionPublishNone
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionPublishNone is implemer
    //::VkUnimplemented ( w, "BbFormat::doOptionPublishNone" );

    _objMag -> msgsRight.publish = PUBLISH_NONE;

    //---- End editable code block: BbFormat doOptionPublishNone
} // End BbFormat::doOptionPublishNone()

void BbFormat::doOptionPublishShear ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOptionPublishShear
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionPublishShear is impleme
    //::VkUnimplemented ( w, "BbFormat::doOptionPublishShear" );

    float mu = 0.035; //blood vicousity varies with Hematocrit
                        // mu = 0.035 <--> H = 0.45 (45%)
    float pi = 3.141592654;
    float r = _radius;
    float tau = 4.0 * fabsf(_avgFlow) * mu / ( 60.0 * pi * pow(r, 3.0) );

```



```

_shear = tau;

char str[200];
sprintf(str, "%6.3f", tau);
XmTextFieldSetString(_textfieldExtension, str);

sprintf(str, "%sindex.html", XmTextFieldGetString(_textfieldNewPath1));
toHTMLFile(str, 1);
sprintf(str, "%sflow3D.html", XmTextFieldGetString(_textfieldNewPath1));
toHTMLFile(str, 2);

//----- End editable code block: BbFormat doOptionPublishShear
} // End BbFormat::doOptionPublishShear()

void BbFormat::doOptionRGB ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionRGB

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionRGB is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionRGB" );

    _format = IMAGE_RGB;

    //----- End editable code block: BbFormat doOptionRGB
} // End BbFormat::doOptionRGB()

void BbFormat::doOptionROI ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionROI

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionROI is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionROI" );

    _type = IMAGE_ROI;

    //----- End editable code block: BbFormat doOptionROI
} // End BbFormat::doOptionROI()

void BbFormat::doOptionTIFF ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbFormat doOptionTIFF

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbFormat::doOptionTIFF is implemented:
    //::VkUnimplemented ( w, "BbFormat::doOptionTIFF" );

    _format = IMAGE_TIFF;

    //----- End editable code block: BbFormat doOptionTIFF
}

```

```
} // End BbFormat::doOptionTIFF()
```

341

```
void BbFormat::doOptionWholeImg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat doOptionWholeImg
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::doOptionWholeImg is implemented
    //::VkUnimplemented ( w, "BbFormat::doOptionWholeImg" );

    _type = IMAGE_WHOLE;

    //---- End editable code block: BbFormat doOptionWholeImg
} // End BbFormat::doOptionWholeImg()
```

```
void BbFormat::newPath ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbFormat newPath
    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;
    //--- Comment out the following line when BbFormat::newPath is implemented:
    ::VkUnimplemented ( w, "BbFormat::newPath" );

    //---- End editable code block: BbFormat newPath
} // End BbFormat::newPath()
```

```
////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////
```

```
VkComponent *BbFormat::CreateBbFormat( const char *name, Widget parent )
{
    VkComponent *obj = new BbFormat ( name, parent );
    return ( obj );
} // End CreateBbFormat
```

```
////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////
```

```
// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details
```

```
struct InterfaceMap {
    char *resourceName;
```

```

char *methodName;
char *argType;
char *definingClass; // Optional, if not this class
void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbFormat::RegisterBbFormatInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int               (Use XmRInt)
    //     float             (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg")
    //     A filename        (Use VkRFilename or "Filename")
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbFormatUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbFormatUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbFormatInterface()

```

```

//---- End of generated code

```

```

//---- Start editable code block: End of generated code

```

```

void BbFormat::init()
{

```

```

    _type = IMAGE_WHOLE;
    _format = IMAGE_RGB;

```

```

    _avgFlow = 0.0;
}

```

```

void BbFormat::setPath(char *path)
{

```

```

    XmTextFieldSetString(_textfieldNewPath1, path);
}

```

```

void BbFormat::savePublish(int type)
{

```

```

char    str[200], str1[200];
int     w1, h1, i;
XImage *x;
Widget  wid;

_objMag -> msgsRight.publish = type;

switch (type)
{
case PUBLISH_2DMAG: sprintf(str1, "mag"); break;
case PUBLISH_2DPHA: sprintf(str1, "pha"); break;
case PUBLISH_2DLOC: sprintf(str1, "loc2D"); break;
case PUBLISH_2DWAVE: sprintf(str1, "wave"); break;
case PUBLISH_3DFLOW:
    if(_type == IMAGE_WHOLE) _type = IMAGE_3DFLOWLARGE;
    i = _objMag -> msgsRight.img_number;
    if(_type == IMAGE_3DFLOWSMALL) sprintf(str1, "flow3D-%d", i);
    else if(_type == IMAGE_3DFLOWLARGE) sprintf(str1, "flow3D-%d", i);
    break;
case PUBLISH_3DLOC:
    if(_type == IMAGE_WHOLE) _type = IMAGE_3DLOCLARGE;
    if(_type == IMAGE_3DLOCSMALL) sprintf(str1, "loc3D");
    else if(_type == IMAGE_3DLOCLARGE) sprintf(str1, "loc3DLarge");
    break;
default: break;
}
sprintf(str, "%s%s", XmTextFieldGetString(_textfieldNewPath1), str1);

switch (type)
{
case PUBLISH_2DMAG:
case PUBLISH_2DPHA:
    w1 = _objMag->_imgView->get_width();
    h1 = _objMag->_imgView->get_height();
    x = _objMag->_imgView->get_XImage();
    break;
case PUBLISH_2DLOC:
    w1 = _objMag->_imgView2->get_width();
    h1 = _objMag->_imgView2->get_height();
    x = _objMag->_imgView2->get_XImage();
    break;
case PUBLISH_2DWAVE:
    w1 = 595;
    h1 = 155;
    wid = _objMag->_bb->baseWidget();
    if( (x = XGetImage(XtDisplay(wid), XtWindow(wid),
                      15, 715, w1, h1, AllPlanes, ZPixmap)) == NULL)
    {
        x = NULL;
    }
    break;
case PUBLISH_3DFLOW:
    wid = _objMag->_bb->baseWidget();
    if(_type == IMAGE_3DFLOWSMALL)
    {
        w1 = 256;
        h1 = 256;
        if( (x = XGetImage(XtDisplay(wid), XtWindow(wid),
                          790, 210, w1, h1, AllPlanes, ZPixmap)) == NULL)
        {
            x = NULL;
        }
    }
    else if(_type == IMAGE_3DFLOWLARGE)
    {
        w1 = 500;
    }
}

```

```

h1 = 500;
if( (x = XGetImage(XtDisplay(wid), XtWindow(wid),
                  670, 90, w1, h1, AllPlanes, ZPixmap)) == NULL)
{
    x = NULL;
}
}
break;
case PUBLISH_3DLOC:
    if(_type == IMAGE_3DLOCSMALL)
    {
        w1 = 256;
        h1 = 256;
    }
    else if(_type == IMAGE_3DLOCLARGE)
    {
        w1 = 600;
        h1 = 700;
    }

    if(_objMag->_win3D != NULL)
    {
        wid = _objMag->_win3D->baseWidget();
        if( (x = XGetImage(XtDisplay(wid), XtWindow(wid),
                          0, 0, w1, h1, AllPlanes, ZPixmap)) == NULL)
        {
            x = NULL;
        }
    }
    else x = NULL;
    break;
default:
    break;
}

if(w1 > 0 && h1 > 0 && x != NULL)
{
    toFile(str, w1, h1, x);
    if(type == PUBLISH_3DLOC && _type == IMAGE_3DLOCLARGE)
    {
        sprintf(str, "%s%s.html", XmTextFieldGetString(_textfieldNewPath1), str1);
        toHTMLFile(str, 3);
    }
    if(type == PUBLISH_3DFLOW)
    {
        char cmd[200];
        sprintf(cmd, "toppm img.rgb %s.ppm", str);
        system(cmd);
    }
    if(type == PUBLISH_2DWAVE)
    {
        _avgFlow = toFlowFile();
        updatePCMR_HTML(fabsf(_avgFlow));
    }
}
}

void BbFormat::updatePCMR_HTML(float avg)
{
    struct pcmrStr {
        char    name[200];
        float   avgFlow;
    } pcmrHTML[15];

    char *strVessel, *strPatient, *strDate, *strRemark;
    strPatient = ((BbVisual *) (_objMag -> _RVis1)) -> get_patient();

```

```

strVessel = ((BbVisual*)(_objMag -> _RVisl)) -> get_vessel();
strDate = ((BbVisual*)(_objMag -> _RVisl)) -> get_date();
strRemark = ((BbVisual*)(_objMag -> _RVisl)) -> get_remark();
time_t t = time(NULL);

int i, i0;
char s[300], str[300];
FILE *fp;

sprintf(s, "%s", XmTextFieldGetString(_textfieldNewPath1));

int n = strlen(s);
for(i=n-2; i>=0; i--)
{
    if(s[i] == '/') {i0 = i; break;}
}

char *anatomy = new char[n-2-i0];
char *pubPath = new char[i0+1];

for(i=i0+1; i<= n-2; i++)
    anatomy[i-i0-1] = s[i];
anatomy[n-2-i0] = '\0';
for(i=0; i<=i0; i++)
    pubPath[i] = s[i];
pubPath[i0+1] = '\0';

sprintf(str, "%s.index", pubPath);
if((fp = fopen(str, "r")) == NULL)
{
    sprintf(pcmrHTML[0].name, "ba");
    sprintf(pcmrHTML[1].name, "laca");
    sprintf(pcmrHTML[2].name, "lcca");
    sprintf(pcmrHTML[3].name, "leca");
    sprintf(pcmrHTML[4].name, "lica-neck");
    sprintf(pcmrHTML[5].name, "lica-intra");
    sprintf(pcmrHTML[6].name, "lmca");
    sprintf(pcmrHTML[7].name, "lva");
    sprintf(pcmrHTML[8].name, "raca");
    sprintf(pcmrHTML[9].name, "rcca");
    sprintf(pcmrHTML[10].name, "reca");
    sprintf(pcmrHTML[11].name, "rica-neck");
    sprintf(pcmrHTML[12].name, "rica-intra");
    sprintf(pcmrHTML[13].name, "rmca");
    sprintf(pcmrHTML[14].name, "rva");
    for(i=0; i<15; i++)
        pcmrHTML[i].avgFlow = -1;
}
else
{
    fp = fopen(str, "r");
    for(i=0; i<15; i++)
        fscanf(fp, "%s %f", pcmrHTML[i].name, &pcmrHTML[i].avgFlow);
    fclose(fp);
}

for(i=0; i<15; i++)
    if(strcmp(anatomy, pcmrHTML[i].name) == 0)
    {
        pcmrHTML[i].avgFlow = avg;
        break;
    }

fp = fopen(str, "w");
for(i=0; i<15; i++)
    fprintf(fp, "%s %f\n", pcmrHTML[i].name, pcmrHTML[i].avgFlow);

```

```

fclose(fp);

sprintf(str, "%sindex.html", pubPath);
printf(" $$$$ $s $$$$ \n", str);

fp = fopen(str, "w");

fprintf(fp, "<HTML>\n");
fprintf(fp, "<BODY>\n");
fprintf(fp, "<H1 ALIGN=CENTER><B>PCMR Blood Flow Report<SUP>*</SUP></B></H1>\n");

fprintf(fp, "<CENTER><P><I>CANVAS Patent Pending</I></P></CENTER>\n");

fprintf(fp, "<CENTER><P><I>Neurosurgery Departement, University of Illinois </I></P></CENTER>\n");

//fprintf(fp, "<H1 ALIGN=CENTER><I><FONT SIZE=-1>%s</FONT></I></H1>\n", asctime(loc
fprintf(fp, "<H1 ALIGN=CENTER><I><FONT SIZE=-1>%s</FONT></I></H1>\n", strDate);

fprintf(fp, "<H1 ALIGN=CENTER>\n");
fprintf(fp, "<HR WIDTH=\"100%%\"></H1>\n");

fprintf(fp, "<UL>\n");
fprintf(fp, "<UL>\n");
fprintf(fp, "<CENTER><P><U>%s</U></P></CENTER>\n", strRemark);

fprintf(fp, "<TABLE BORDER=3 CELLSPACING=10 CELLPADDING=0 >\n");
fprintf(fp, "<TR ALIGN=LEFT>\n");
fprintf(fp, "<TD><B>Patient Name</B></TD>\n");

fprintf(fp, "<TD><I>%s</I></TD>\n", strPatient);
fprintf(fp, "</TR>\n");

fprintf(fp, "<TR>\n");
fprintf(fp, "<TD>Requested \n");
fprintf(fp, "<P>Vessels</P>\n");

//fprintf(fp, "<P>%s</P>\n", strDate);
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
fprintf(fp, "<UL>\n");
if(pcmrHTML[0].avgFlow >= 0)
    fprintf(fp, "<LI><A HREF=\"../ba/index.html\">BA</A> <I>: %7.2f &nbsp;&nbsp;&nbsp;</I></LI></");
else
    fprintf(fp, "<LI>BA<I>: &nbsp;&nbsp;&nbsp;</I></LI>\n");
fprintf(fp, "</UL>\n");

fprintf(fp, "<UL>\n");
fprintf(fp, "<LI><I>Left Side &nbsp;&nbsp;&nbsp;</I></LI>\n");

fprintf(fp, "<TABLE BORDER=4 CELLSPACING=3 CELLPADDING=3 >\n");
fprintf(fp, "<TR>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[1].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"../laca/index.html\">ACA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>ACA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[2].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"../lcca/index.html\">CCA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>CCA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

```

```

fprintf(fp, "<TD>\n");
if(pcmrHTML[3].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./leca/index.html\">ECA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>ECA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[4].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./lica-neck/index.html\">ICA-Neck</A></P></CENT");
else
    fprintf(fp, "<CENTER><P>ICA-Neck</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[5].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./lica-intra/index.html\">ICA-Intra</A></P></CE");
else
    fprintf(fp, "<CENTER><P>ICA-Intra</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[6].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./lmca/index.html\">MCA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>MCA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[7].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./lva/index.html\">VA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>VA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TR>\n");

for(i=1; i<=7; i++)
    if(pcmrHTML[i].avgFlow >= 0)
    {
        fprintf(fp, "<TD>\n");
        fprintf(fp, "<CENTER><P>%7.2f</P></CENTER>\n", pcmrHTML[i].avgFlow);
        fprintf(fp, "</TD>\n");
    }
    else fprintf(fp, "<TD></TD>\n");

fprintf(fp, "</TR>\n");
fprintf(fp, "</TABLE>\n");
fprintf(fp, "</UL>\n");

fprintf(fp, "<UL>\n");
fprintf(fp, "<LI><I>Right Side</I></LI>\n");

fprintf(fp, "<TABLE BORDER=4 CELLSPACING=3 CELLPADDING=3 >\n");
fprintf(fp, "<TR>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[8].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./raca/index.html\">ACA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>ACA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[9].avgFlow >= 0)

```



```

    fprintf(fp, "<CENTER><P><A HREF=\"./rcca/index.html\">CCA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>CCA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[10].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./reca/index.html\">ECA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>ECA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[11].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./rica-neck/index.html\">ICA-Neck</A></P></CENT
else
    fprintf(fp, "<CENTER><P>ICA-Neck</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[12].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./rica-intra/index.html\">ICA-Intra</A></P></CE
else
    fprintf(fp, "<CENTER><P>ICA-Intra</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[13].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./rmca/index.html\">MCA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>MCA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TD>\n");
if(pcmrHTML[14].avgFlow >= 0)
    fprintf(fp, "<CENTER><P><A HREF=\"./rva/index.html\">VA</A> </P></CENTER>\n");
else
    fprintf(fp, "<CENTER><P>VA</P></CENTER>\n");
fprintf(fp, "</TD>\n");

fprintf(fp, "<TR>\n");

for(i=8; i<=14; i++)
    if(pcmrHTML[i].avgFlow >= 0)
    {
        fprintf(fp, "<TD>\n");
        fprintf(fp, "<CENTER><P>%7.2f</P></CENTER>\n", pcmrHTML[i].avgFlow);
        fprintf(fp, "</TD>\n");
    }
    else fprintf(fp, "<TD></TD>\n");

fprintf(fp, "</TR>\n");
fprintf(fp, "</TABLE>\n");
fprintf(fp, "</UL>\n");
fprintf(fp, "</TD>\n");
fprintf(fp, "</TR>\n");
fprintf(fp, "</TABLE>\n");

fprintf(fp, "<P><FONT SIZE=+2>* </FONT>For Information Exchange Only, Further Clini
fprintf(fp, "Correlation is Needed. For better results, (1) Good Quality and digit
fprintf(fp, "data of PCMR sent to CANVAS workstation; (2) Phase Contrast MR Flow St
fprintf(fp, "performed by Using CANVAS Protocol.</P>\n");
fprintf(fp, "</UL>\n");
fprintf(fp, "</UL>\n");

```

```

    fprintf(fp, "</BODY>\n");
    fprintf(fp, "</HTML>\n");
    fclose(fp);
}

float BbFormat::toFlowFile()
{
    char fname[300];
    sprintf(fname, "%sflow.dat", XmTextFieldGetString(_textfieldNewPath1));

    char *strVessel, *strPatient, *strDate, *strRemark;
    int vessel = _objMag -> _vessel;

    strPatient = ((BbVisual *) (_objMag -> _RVisl)) -> get_patient();
    strVessel = ((BbVisual *) (_objMag -> _RVisl)) -> get_vessel();
    strDate = ((BbVisual *) (_objMag -> _RVisl)) -> get_date();
    strRemark = ((BbVisual *) (_objMag -> _RVisl)) -> get_remark();

    FILE *fp = fopen(fname, "w");

    fprintf(fp, "\n*****\nPatient: %s\n", strPatient);
    fprintf(fp, "Anatomy: %s\n", strVessel);
    fprintf(fp, "Study Date: %s\n", strDate);
    fprintf(fp, "Remark: %s\n", strRemark);

    time_t t = time(NULL);
    fprintf(fp, "Analysis Date: %s\n", asctime(localtime(&t)));

    fprintf(fp, "\nCANVAS_PCMRA_FLOW_97\n");
    fprintf(fp, "%s\n", strVessel);
    fprintf(fp, "%d\n", int(_objMag->msgsRight.HR));
    fprintf(fp, "%d\n", _objMag->msgsRight.num_imgs);

    float avg = 0;

    for(int i=0; i<_objMag->msgsRight.num_imgs; i++)
    {
        fprintf(fp, "%d %f %f %f %f %f\n", i+1, _objMag -> _flow[vessel].vesselFlows[i].v
            _objMag -> _flow[vessel].vesselFlows[i].psv, _objMag -> _flow[vessel].vesselFlows[
            _objMag -> _flow[vessel].vesselFlows[i].mv, _objMag -> _flow[vessel].vesselFlows[i
            avg += _objMag -> _flow[vessel].vesselFlows[i].vfr;
    }

    avg /= float(_objMag->msgsRight.num_imgs);
    fprintf(fp, "\n\nAverage Flow Rate: %f mL/min\n", avg);

    fclose(fp);

    return avg;
}

void BbFormat::toHTMLFile(char *fname, int type)
{
    int i;
    char *strVessel, *strPatient, *strDate, *strRemark;

    strPatient = ((BbVisual *) (_objMag -> _RVisl)) -> get_patient();
    strVessel = ((BbVisual *) (_objMag -> _RVisl)) -> get_vessel();
    strDate = ((BbVisual *) (_objMag -> _RVisl)) -> get_date();
    strRemark = ((BbVisual *) (_objMag -> _RVisl)) -> get_remark();

    FILE *fp = fopen(fname, "w");

    fprintf(fp, "<HTML>\n<BODY>\n");
    fprintf(fp, "<H1 ALIGN=CENTER>PCMR &quot;%s&quot; Flow</H1>\n", strVessel);
    fprintf(fp, "<P><B>Patient Name: </B>%s\n", strPatient);

```

```
fprintf(fp, "      \n");  
fprintf(fp, "<B>Date:</B>\n", strDate);  
fprintf(fp, "\n");  
fprintf(fp, "<U>%s</U></P>\n", strRemark);  
fprintf(fp, "<P><HR WIDTH=\"100%%\"></P>\n");
```

350

```
switch(type)
```

case 1:

```
fprintf(fp, "<CENTER><P>Diameter: %7.3f &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Shear: %7.3f\n");
fprintf(fp, "<CENTER><P><A HREF=\"flow3D.html\"><IMG SRC=\"wave.gif\" BORDER=0\">\n");
fprintf(fp, "HEIGHT=155 WIDTH=595</A></P></CENTER>\n");
fprintf(fp, "<CENTER><P><IMG SRC=\"mag.gif\" HEIGHT=256 WIDTH=256>);\n");
fprintf(fp, "    <IMG SRC=\"pha.gif\" HEIGHT=256 WIDTH=256></P></CENTER>\n");
fprintf(fp, "<CENTER><P><IMG SRC=\"loc2D.gif\" HEIGHT=256 WIDTH=256>);\n");
fprintf(fp, "    <A HREF=\"loc3DLarge.html\"><IMG SRC=\"loc3D.gif\" BORDER=0\">\n");
fprintf(fp, "HEIGHT=256 WIDTH=256</A></P></CENTER>\n");
break;
```

case 2:

```
for(i=0; i<_objMag->msgsRight.num_imgs; i += 4)
```

```

    fprintf(fp, "<CENTER><P><IMG SRC=\"flow3D-%d.gif\" HEIGHT=128 WIDTH=128>", i+1);
    fprintf(fp, "<IMG SRC=\"flow3D-%d.gif\" HEIGHT=128 WIDTH=128>", i+2);
    fprintf(fp, "<IMG SRC=\"flow3D-%d.gif\" HEIGHT=128 WIDTH=128>", i+3);
    fprintf(fp, "<IMG SRC=\"flow3D-%d.gif\" HEIGHT=128 WIDTH=128></P></CENTER>\n"
}

```

```
fprintf(fp, "<CENTER><P><A HREF=\"flow3D.mpg\">3D Flow Movie</A></P></CENTER>\n");
break;
```

case 3:

```
fprintf(fp, "<CENTER><P><IMG SRC=\"loc3DLarge.gif\" HEIGHT=700 WIDTH=600></P></>");
break;
```

```
default:
```

```
break;
```

```
fprintf(fp, "</BODY>\n</HTML>\n");
fclose(fp);
```

```
void BbFormat::toFile(char *fname, int w, int h, XImage *ximage)
```

```
{
    FILE *fp = fopen("img.bin", "w");
    unsigned char *pp = (unsigned char *) malloc (1);
    int    x, y, pos;
    char    cmd[200];
```

```

/*
for(x=h-1; x>=0; x--)
for(y=0; y<w; y++)
{
    pos = (x*w + y) * 4;
    if(ximage -> data[pos+1] < 30 && ximage -> data[pos+2] < 30
        && ximage -> data[pos+3] < 30)
    {
        ximage -> data[pos+1] = 30;
        ximage -> data[pos+2] = 30;
        ximage -> data[pos+3] = 30;
    }
}
*/

```

```
for(x=h-1; x>=0; x--)
for(y=0; y<w; y++) {
    pos = (x*w + y) * 4;
    *pp = ximage -> data[pos+3];
    fwrite(pp, 1, 1, fp);
}
```

```

    }
    for(x=h-1; x>=0; x--)
    for(y=0; y<w; y++) {
        pos = (x*w + y) * 4;
        *pp = ximage -> data[pos+2];
        fwrite(pp, 1, 1, fp);
    }
    for(x=h-1; x>=0; x--)
    for(y=0; y<w; y++) {
        pos = (x*w + y) * 4;
        *pp = ximage -> data[pos+1];
        fwrite(pp, 1, 1, fp);
    }
    fclose(fp);

    sprintf(cmd, "frombin img.bin img.rgb %d %d 3", w, h);
    system(cmd);
    sprintf(cmd, "togif img.rgb %s.gif", fname);
    system(cmd);
}

void BbFormat::toMPEG()
{
    char    str[300], cmd[300], p[300];
    FILE    *fp;
    int     n = _objMag->msgsRight.num_imgs;

    sprintf(p, "%s", XmTextFieldGetString(_textfieldNewPath1));
    for(int i=1; i<=n; i++)
    {
        sprintf(cmd, "cp %sflow3D-%d.ppm %sflow3D-%d.ppm", p, i, p, n+i);
        system(cmd);
        sprintf(cmd, "cp %sflow3D-%d.ppm %sflow3D-%d.ppm", p, i, p, 2*n+i);
        system(cmd);
    }

    sprintf(str, "%smpeg.param", p);

    fp = fopen(str, "w");
    fprintf(fp, "PATTERN          IBBPBBPBBPBBPBBP\n");

    fprintf(fp, "OUTPUT          %sflow3D.mpg\n", p);

    fprintf(fp, "BASE_FILE_FORMAT PPM\n");
    fprintf(fp, "INPUT_CONVERT    *\n");
    fprintf(fp, "GOP_SIZE 16\n");
    fprintf(fp, "SLICES_PER_FRAME 1\n");

    fprintf(fp, "INPUT_DIR        %s\n", p);

    fprintf(fp, "INPUT\n");

    fprintf(fp, "flow3D-*.ppm     [1-%d]\n", n*3);

    fprintf(fp, "END_INPUT\n");
    fprintf(fp, "# FULL or HALF -- must be upper case\n");
    fprintf(fp, "PIXEL           HALF\n");
    fprintf(fp, "RANGE           10\n");
    fprintf(fp, "# this must be one of {EXHAUSTIVE, SUBSAMPLE, LOGARITHMIC}\n");
    fprintf(fp, "PSEARCH_ALG     LOGARITHMIC\n");
    fprintf(fp, "BSEARCH_ALG     CROSS2\n");
    fprintf(fp, "IQSCALE         8\n");
    fprintf(fp, "PQSCALE         10\n");
    fprintf(fp, "BQSCALE         25\n");
    fprintf(fp, "REFERENCE_FRAME ORIGINAL\n");
    fprintf(fp, "# The frame rate is the number of frames/second (legal values:\n");

```

```

fprintf(fp, "# 23.976, 24, 25, 29.97, 30, 50, 59.94, 60");

float fr = n * _objMag->msgsRight.HR / 60.0;

if(fr <= 23.976) fr = 23.976;
else if(fr < 24.5) fr = 24;
else if(fr < 27.5) fr = 25;
else if(fr <= 29.97) fr = 29.97;
else if(fr < 40) fr = 30;
else if(fr < 55) fr = 50;
else if(fr <= 59.94) fr = 59.94;
else fr = 60;

fprintf(fp, "FRAME_RATE %f\n", fr);

fclose(fp);

sprintf(cmd, "/usr/people/meide/TOOLS/mpeg/mpeg_encode/mpeg_encode %s", str);
system(cmd);
sprintf(cmd, "rm %s*.ppm", p);
system(cmd);
}

void BbFormat::toFile(char *fname)
{
    char str[100];
    FILE *fp = fopen("img.bin", "w");
    unsigned char *pp = (unsigned char *) malloc (1);

    int w, h;
    float zoom;

    if(_type == IMAGE_WHOLE)
    {
        w = _objMag -> _imgView2 -> get_width();
        h = _objMag -> _imgView2 -> get_height();
    }
    else if(_type == IMAGE_ROI)
    {
        zoom = 1.0;
        //zoom = _objMag -> _imgView2 -> _zoom;
        w = int(_objMag -> _imgView2 -> get_width()/zoom);
        h = int(_objMag -> _imgView2 -> get_height()/zoom);
    }

    for(int x=h-1; x>=0; x--)
    for(int y=0; y<w; y++) {
        if(_type == IMAGE_WHOLE)
            *pp = _objMag -> _imgView2 -> _cimg-> red[x][y];
        else if(_type == IMAGE_ROI)
            *pp = _objMag -> _imgView2 -> _cimg-> red[int(x*zoom)][int(y*zoom)];
        *pp = 255 - *pp;
        fwrite(pp, 1, 1, fp);
    }
    fclose(fp);

    if( _objMag->_imgView2->_ROI != NULL)
    {
        sprintf(str, "%s.con", fname);
        fp = fopen(str, "w");
        fprintf(fp, "1010\n1\n");
        Points *p = _objMag->_imgView2->_ROI->_points_in_border.get_Points(zoom, 0, 0);
        p -> to_ContourFile(fp);
        fprintf(fp, "1.0\n");
        fclose(fp);
    }
}

```

```
}  
if(_format == IMAGE_BIN) return;  
else  
{  
    char cmd[300];  
  
    sprintf(cmd, "frombin img.bin img.rgb %d %d", w, h);  
    system(cmd);  
  
    if(_format == IMAGE_RGB)  
        sprintf(cmd, "mv img.rgb %s", fname);  
    else  
    {  
        if(_format == IMAGE_GIF)  
            sprintf(cmd, "togif img.rgb %s", fname);  
        else  
            sprintf(cmd, "mv img.rgb %s", fname);  
    }  
    system(cmd);  
}  
}  
  
//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbFormatUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbFormatUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbFormatUI::_defaultBbFormatUIResources[] = {
    "*button3DContour.labelString: 3D Contour",
    "*buttonAcceptFlow1.labelString: Accept ",
    "*buttonPublishPath.labelString: DIR",
    "*buttonShow2DContour.labelString: 2D Contour",
    "*labelFname.labelString: Diameter",
    "*labelFnameExt.labelString: Shear",
    "*option2DLoc.labelString: 2D Loc",
    "*option2DMag.labelString: 2D Mag",
    "*option2DPhase.labelString: 2D Phase",
    "*option2DWave.labelString: 2D Wave",
    "*option3DFlow.labelString: 3D Flow",
    "*option3DFlowLarge.labelString: 3D Flow Large",
    "*option3DFlowSmall.labelString: 3D Flow Small",
    "*option3DLoc.labelString: 3D Loc",
    "*option3DLocLarge.labelString: 3D Loc Large",
    "*option3DLocSmall.labelString: 3D Loc Small",
    "*optionGIF.labelString: GIF",
    "*optionHTML.labelString: HTML File",
    "*optionMPEG.labelString: MPEG File",
    "*optionPublishArea.labelString: Area",
    "*optionPublishNone.labelString: None",

```

```

    *optionPublishShear.labelString:  Shear",
    *optionRGB.labelString:  RGB",
    *optionROI3.labelString:  ROI",
    *optionTIFF.labelString:  TIFF",
    *optionWholeImg.labelString:  Whole",
    *tabLabel:  Publish",
    *textfieldNewPath1.value:  /usr/people/meide/images/tmp",

```

```

//---- Start editable code block: BbFormatUI Default Resources

```

```

//---- End editable code block: BbFormatUI Default Resources

```

```

(char*)NULL

```

```

};

```

```

BbFormatUI::BbFormatUI ( const char *name ) : VkComponent ( name )
{

```

```

    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

```

```

//---- Start editable code block: BbFormat constructor 2

```

```

//---- End editable code block: BbFormat constructor 2

```

```

}    // End Constructor

```

```

BbFormatUI::BbFormatUI ( const char *name, Widget parent ) : VkComponent ( name )
{

```

```

    //---- Start editable code block: BbFormat pre-create

```

```

//---- End editable code block: BbFormat pre-create

```

```

// Call creation function to build the widget tree.

```

```

    create ( parent );

```

```

//---- Start editable code block: BbFormat constructor

```

```

//---- End editable code block: BbFormat constructor

```

```

}    // End Constructor

```

```

BbFormatUI::~BbFormatUI()
{

```

```

    // Base class destroys widgets

```

```

//---- Start editable code block: BbFormatUI destructor

```

```

//---- End editable code block: BbFormatUI destructor

```

```

}    // End destructor

```


[illegible]

```

_option2DMag = _optionMenu16->addAction ( "option2DMag",
&BbFormatUI::doOption2DMagCallback,
(XtPointer) this );

_option2DPhase = _optionMenu16->addAction ( "option2DPhase",
&BbFormatUI::doOption2DPhaseCallback,
(XtPointer) this );

_option2DWave = _optionMenu16->addAction ( "option2DWave",
&BbFormatUI::doOption2DWaveCallback,
(XtPointer) this );

_option2DLoc = _optionMenu16->addAction ( "option2DLoc",
&BbFormatUI::doOption2DLocCallback,
(XtPointer) this );

_option3DLoc = _optionMenu16->addAction ( "option3DLoc",
&BbFormatUI::doOption3DLocCallback,
(XtPointer) this );

_option3DFlow = _optionMenu16->addAction ( "option3DFlow",
&BbFormatUI::doOption3DFlowCallback,
(XtPointer) this );

_optionHTML = _optionMenu16->addAction ( "optionHTML",
&BbFormatUI::doOptionHTMLCallback,
(XtPointer) this );

_optionMPEG = _optionMenu16->addAction ( "optionMPEG",
&BbFormatUI::doOptionMPEGCallback,
(XtPointer) this );

_optionPublishArea = _optionMenu16->addAction ( "optionPublishArea",
&BbFormatUI::doOptionPublishAreaCallback,
(XtPointer) this );

_optionPublishShear = _optionMenu16->addAction ( "optionPublishShear",
&BbFormatUI::doOptionPublishShearCallback,
(XtPointer) this );

_button3DContour = XtVaCreateManagedWidget ( "button3DContour",
XmPushButtonWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 360,
XmNy, 109,
XmNwidth, 110,
XmNheight, 30,
(XtPointer) NULL );

XtAddCallback ( _button3DContour,
XmNactivateCallback,
&BbFormatUI::doButton3DContourCallback,
(XtPointer) this );

_buttonShow2DContour = XtVaCreateManagedWidget ( "buttonShow2DContour",
XmPushButtonWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 360,
XmNy, 65,
XmNwidth, 110,
XmNheight, 30,
(XtPointer) NULL );

```

[illegible]

```

        _baseWidget,
        XmNLabelType, XmSTRING,
        XmNx, 360,
        XmNy, 19,
        XmNwidth, 110,
        XmNheight, 30,
        (XtPointer) NULL );

XtAddCallback ( _buttonAcceptFlow1,
                XmNactivateCallback,
                &BbFormatUI::doButtonAcceptFlowCallback,
                (XtPointer) this );

_labelFname = XtVaCreateManagedWidget ( "labelFname",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 511,
                                         XmNy, 10,
                                         XmNwidth, 60,
                                         XmNheight, 20,
                                         (XtPointer) NULL );

_labelFnameExt = XtVaCreateManagedWidget ( "labelFnameExt",
                                             xmLabelWidgetClass,
                                             _baseWidget,
                                             XmNlabelType, XmSTRING,
                                             XmNx, 517,
                                             XmNy, 80,
                                             XmNwidth, 45,
                                             XmNheight, 20,
                                             (XtPointer) NULL );

XtVaSetValues ( _optionMenu16->baseWidget(),
                XmNx, 22,
                XmNy, 104,
                XmNwidth, 132,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenu13->baseWidget(),
                XmNx, 176,
                XmNy, 104,
                XmNwidth, 158,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenu9->baseWidget(),
                XmNx, 239,
                XmNy, 10,
                XmNwidth, 92,
                XmNheight, 32,
                (XtPointer) NULL );

//---- Start editable code block: BbFormatUI create

//---- End editable code block: BbFormatUI create
}

const char * BbFormatUI::className()
{
    return ("BbFormatUI");
} // End className()

```

[illegible]


```

    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionHTML ( w, callData );
}

void BbFormatUI::doOptionMPEGCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionMPEG ( w, callData );
}

void BbFormatUI::doOptionPublishAreaCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionPublishArea ( w, callData );
}

void BbFormatUI::doOptionPublishNoneCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionPublishNone ( w, callData );
}

void BbFormatUI::doOptionPublishShearCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionPublishShear ( w, callData );
}

void BbFormatUI::doOptionRGBCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionRGB ( w, callData );
}

void BbFormatUI::doOptionROICallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionROI ( w, callData );
}

void BbFormatUI::doOptionTIFFCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionTIFF ( w, callData );
}

void BbFormatUI::doOptionWholeImgCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->doOptionWholeImg ( w, callData );
}

```

```

}

void BbFormatUI::newPathCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbFormatUI* obj = ( BbFormatUI * ) clientData;
    obj->newPath ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbFormatUI::doButton3DContour ( Widget, XtPointer )
{
    // This virtual function is called from doButton3DContourCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doButtonAcceptFlow ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptFlowCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doButtonPublishPath ( Widget, XtPointer )
{
    // This virtual function is called from doButtonPublishPathCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doButtonShow2DContour ( Widget, XtPointer )
{
    // This virtual function is called from doButtonShow2DContourCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption2DLoc ( Widget, XtPointer )
{
    // This virtual function is called from doOption2DLocCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption2DMag ( Widget, XtPointer )
{
    // This virtual function is called from doOption2DMagCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption2DPhase ( Widget, XtPointer )
{
    // This virtual function is called from doOption2DPhaseCallback.
    // This function is normally overridden by a derived class.
}

```



```

void BbFormatUI::doOption2Wave ( Widget, XtPointer )
{
    // This virtual function is called from doOption2DWaveCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DFlow ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DFlowCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DFlowLarge ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DFlowLargeCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DFlowSmall ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DFlowSmallCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DLoc ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DLocCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DLocLarge ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DLocLargeCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOption3DLocSmall ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DLocSmallCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionGIF ( Widget, XtPointer )
{
    // This virtual function is called from doOptionGIFCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionHTML ( Widget, XtPointer )
{
    // This virtual function is called from doOptionHTMLCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionMPEG ( Widget, XtPointer )
{
    // This virtual function is called from doOptionMPEGCallback.

```

```

    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionPublishArea ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPublishAreaCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionPublishNone ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPublishNoneCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionPublishShear ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPublishShearCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionRGB ( Widget, XtPointer )
{
    // This virtual function is called from doOptionRGBCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionROI ( Widget, XtPointer )
{
    // This virtual function is called from doOptionROICallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionTIFF ( Widget, XtPointer )
{
    // This virtual function is called from doOptionTIFFCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::doOptionWholeImg ( Widget, XtPointer )
{
    // This virtual function is called from doOptionWholeImgCallback.
    // This function is normally overridden by a derived class.
}

void BbFormatUI::newPath ( Widget, XtPointer )
{
    // This virtual function is called from newPathCallback.
    // This function is normally overridden by a derived class.
}

```

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

User: meide
Host: phoenix
Class: phoenix
Job: BbFlowUI.C

364

```

////////////////////////////////////
//
// Source file for BbHistogram
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbHistogramUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbHistogram.h"
#include <Vk/VkEZ.h>
#include <Sgm/Dial.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbHistogramUI and are
// available as protected data members inherited by this class
//
// VkOptionMenu *          _optionMenuLHist
// VkMenuItem *           _optionLUpdate
// VkMenuItem *           _optionLCoarse
// VkMenuItem *           _optionLFine
// VkMenuItem *           _optionLMapping
// SgDial                  _dialCenter
// SgDial                  _dialWidth
// VkOptionMenu *          _optionMenuLHistogram
// VkMenuItem *           _optionHead
// VkMenuItem *           _optionBone
// VkMenuItem *           _optionLung
// VkMenuItem *           _optionSpine
// VkMenuItem *           _optionAbdomen
// VkMenuItem *           _optionMediaStinum
// XmLabel                  _labelLHistoMax
// XmLabel                  _labelLHistoHigh
// XmLabel                  _labelLHistoLow
// XmLabel                  _labelLHistoMin
//
////////////////////////////////////

///---- Start editable code block: headers and declarations

#include "Utility.h"
#include <math.h>

```

//---- BbHistogram Constructor

```
BbHistogram::BbHistogram(const char *name, Widget parent) :  
    BbHistogramUI(name, parent)
```

```
{  
    // This constructor calls BbHistogramUI(parent, name)  
    // which calls BbHistogramUI::create() to create  
    // the widgets for this component. Any code added here  
    // is called after the component's interface has been built  
  
    //---- Start editable code block: BbHistogram constructor  
  
    init();  
    //---- End editable code block: BbHistogram constructor  
  
}    // End Constructor
```

```
BbHistogram::BbHistogram(const char *name) :  
    BbHistogramUI(name)
```

```
{  
    // This constructor calls BbHistogramUI(name)  
    // which does not create any widgets. Usually, this  
    // constructor is not used  
  
    //---- Start editable code block: BbHistogram constructor 2  
  
    init();  
    //---- End editable code block: BbHistogram constructor 2  
  
}    // End Constructor
```

```
BbHistogram::~BbHistogram()
```

```
{  
    // The base class destructors are responsible for  
    // destroying all widgets and objects used in this component.  
    // Only additional items created directly in this class  
    // need to be freed here.  
  
    //---- Start editable code block: BbHistogram destructor  
  
    //---- End editable code block: BbHistogram destructor  
  
}    // End Destructor
```

```
const char * BbHistogram::className() // classname  
{  
    return ("BbHistogram");  
} // End className()
```

```

void BbHistogram::abdomen (Widget w, XtPointer callData)
{
    //---- Start editable code block: BbHistogram abdomen

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::abdomen is implemented:

    ::VkUnimplemented ( w, "BbHistogram::abdomen" );


    //---- End editable code block: BbHistogram abdomen
}

// End BbHistogram::abdomen()


void BbHistogram::bone ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram bone

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::bone is implemented:

    ::VkUnimplemented ( w, "BbHistogram::bone" );


    //---- End editable code block: BbHistogram bone
}

// End BbHistogram::bone()


void BbHistogram::centerDrag ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram centerDrag

    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::centerDrag is implemented:

    //::VkUnimplemented ( w, "BbHistogram::centerDrag" );


    //---- End editable code block: BbHistogram centerDrag
}

// End BbHistogram::centerDrag()


void BbHistogram::doOptionCoarse ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram doOptionCoarse

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::doOptionCoarse is implemented:

    //::VkUnimplemented ( w, "BbHistogram::doOptionCoarse" );

    _objMag -> update_Lhisto();
}

```

```

//---- End editable code block: BbHistogram doOptionCoarse
} // End BbHistogram::doOptionCoarse()

void BbHistogram::doOptionLFine ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram doOptionLFine
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::doOptionLFine is implemented
    //::VkUnimplemented ( w, "BbHistogram::doOptionLFine" );
    _objMag -> update_Lhisto2();

    //---- End editable code block: BbHistogram doOptionLFine
} // End BbHistogram::doOptionLFine()

void BbHistogram::doOptionMapping ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram doOptionMapping
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::doOptionMapping is implement
    //::VkUnimplemented ( w, "BbHistogram::doOptionMapping" );
    _objMag -> update_Lhisto2();

    //---- End editable code block: BbHistogram doOptionMapping
} // End BbHistogram::doOptionMapping()

void BbHistogram::doOptionUpdate ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram doOptionUpdate
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::doOptionUpdate is implemented
    //::VkUnimplemented ( w, "BbHistogram::doOptionUpdate" );

    //---- End editable code block: BbHistogram doOptionUpdate
} // End BbHistogram::doOptionUpdate()

void BbHistogram::head ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram head
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

```

```

//--- Comment out the following line when BbHistogram::head is implemented:
::VkUnimplemented ( w, "BbHistogram::head" );

//---- End editable code block: BbHistogram head
} // End BbHistogram::head()

void BbHistogram::highChg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram highChg
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::highChg is implemented:
    //::VkUnimplemented ( w, "BbHistogram::highChg" );

    int tmp;
    SgDialGetValue(w, &tmp);
    if(_winWidth == -1)
        _winWidth = tmp;
    else
    {
        int x = tmp - _winWidth;
        if(fabsf(x) < 200)
            update_width(x);
        _winWidth = tmp;
    }

    //---- End editable code block: BbHistogram highChg
} // End BbHistogram::highChg()

void BbHistogram::lowChg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram lowChg
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;

    //--- Comment out the following line when BbHistogram::lowChg is implemented:
    //::VkUnimplemented ( w, "BbHistogram::lowChg" );

    int tmp;
    SgDialGetValue(w, &tmp);
    if(_winCenter == -1)
        _winCenter = tmp;
    else
    {
        int x = tmp - _winCenter;
        if(fabsf(x) < 200)
            update_center(x);
        _winCenter = tmp;
    }

    //---- End editable code block: BbHistogram lowChg
}

```



```

void BbHistogram::lung ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram lung
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::lung is implemented:
    ::VkUnimplemented ( w, "BbHistogram::lung" );

    //---- End editable code block: BbHistogram lung
} // End BbHistogram::lung()

void BbHistogram::mediastinum ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram mediastinum
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::mediastinum is implemented:
    ::VkUnimplemented ( w, "BbHistogram::mediastinum" );

    //---- End editable code block: BbHistogram mediastinum
} // End BbHistogram::mediastinum()

void BbHistogram::spine ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram spine
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::spine is implemented:
    ::VkUnimplemented ( w, "BbHistogram::spine" );

    //---- End editable code block: BbHistogram spine
} // End BbHistogram::spine()

void BbHistogram::widthDrag ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbHistogram widthDrag
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;
    //--- Comment out the following line when BbHistogram::widthDrag is implemented:

```

```
//---- End editable code block: BbHistogram widthDrag  
}  
// End BbHistogram::widthDrag()
```

```
////////////////////////////////////  
// static creation function, for importing class into rapidapp  
// or dynamically loading, using VkComponent::loadComponent  
////////////////////////////////////
```

```
VkComponent *BbHistogram::CreateBbHistogram( const char *name, Widget parent )  
{  
    VkComponent *obj = new BbHistogram ( name, parent );  
    return ( obj );  
} // End CreateBbHistogram
```

```
////////////////////////////////////  
// Function for accessing a description of the dynamic interface  
// to this class.  
////////////////////////////////////
```

```
// WARNING: This structure is different than that used with 1.1 RapidApp.  
// See the RapidApp release notes for details
```

```
struct InterfaceMap {  
    char *resourceName;  
    char *methodName;  
    char *argType;  
    char *definingClass; // Optional, if not this class  
    void (VkCallbackObject::*method)(...); // Reserved, do not set  
};
```

```
void *BbHistogram::RegisterBbHistogramInterface()  
{  
    // This structure registers information about this class  
    // that allows RapidApp to create and manipulate an instance.  
    // Each entry provides a resource name that will appear in the  
    // resource manager palette when an instance of this class is  
    // selected, the name of the member function as a string,  
    // the type of the single argument to this function, and an.  
    // optional argument indicating the class that defines this function.  
    // All member functions must have the form  
    //  
    // void memberFunction ( Type );  
    //  
    // where "Type" is one of:  
    // const char * (Use XmRString)  
    // Boolean (Use XmRBoolean)  
    // int (Use XmRInt)  
    // float (Use XmRFloat)  
    // No argument (Use VkRNoArg or "NoArg")  
    // A filename (Use VkRFilename or "Filename")  
    // An enumeration (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")  
    // A callback (Use XmRCallback)
```

```

static InterfaceMap m[] = {
//---- Start editable code block: BbHistogramUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbHistogramUI resource table
    { NULL }, // MUST be NULL terminated
};

    return map;
} // End RegisterBbHistogramInterface()

//---- End of generated code

//---- Start editable code block: End of generated code

void BbHistogram::init()
{
    _winCenter = -1;
    _winWidth = -1;
}

void BbHistogram::update_width(int x)
{
    _objMag -> msgsLeft.img_winCenter -= float(x)/2.0;
    _objMag -> msgsLeft.img_winWidth += float(x)/2.0;
    _objMag -> update_LimgView(_objMag -> msgsLeft.img_winCenter,
        _objMag -> msgsLeft.img_winWidth);
}

void BbHistogram::update_center(int x)
{
    _objMag -> msgsLeft.img_winCenter += float(x);
    _objMag -> msgsLeft.img_winWidth += float(x);
    _objMag -> update_LimgView(_objMag -> msgsLeft.img_winCenter,
        _objMag -> msgsLeft.img_winWidth);
}
//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbHistogramUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbHistogramUI.h" // Generated header file for this class

#include <Sgm/Dial.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbHistogramUI::_defaultBbHistogramUIResources[] = {
    "*labelLHistoHigh.labelString: 1",
    "*labelLHistoLow.labelString: 0",
    "*labelLHistoMax.labelString: 1",
    "*labelLHistoMin.labelString: 0",
    "*optionAbdomen.labelString: Abdomen",
    "*optionBone.labelString: Bone",
    "*optionHead.labelString: Head",
    "*optionLCoarse.labelString: Coarse",
    "*optionLFine.labelString: Fine",
    "*optionLMapping.labelString: Mapping",
    "*optionLUpdate.labelString: Update",
    "*optionLung.labelString: Lung",
    "*optionMediaStinum.labelString: MediaStinum",
    "*optionMenuLHist.labelString: ",
    "*optionMenuLHistogram.labelString: ",
    "*optionSpine.labelString: Spine",
    "*tabLabel: View",

    //---- Start editable code block: BbHistogramUI Default Resources

    //---- End editable code block: BbHistogramUI Default Resources

```

};

```
BbHistogramUI::BbHistogramUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //----- Start editable code block: BbHistogram constructor 2

    //----- End editable code block: BbHistogram constructor 2

}    // End Constructor
```

```
BbHistogramUI::BbHistogramUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //----- Start editable code block: BbHistogram pre-create

    //----- End editable code block: BbHistogram pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //----- Start editable code block: BbHistogram constructor

    //----- End editable code block: BbHistogram constructor

}    // End Constructor
```

```
BbHistogramUI::~BbHistogramUI()
{
    // Base class destroys widgets

    //----- Start editable code block: BbHistogramUI destructor

    //----- End editable code block: BbHistogramUI destructor
}    // End destructor
```

```
void BbHistogramUI::create ( Widget parent )
{
    Arg      args[7];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object

    setDefaultResources ( parent, _defaultBbHistogramUIResources );
```

```
// Create an unmanaged widget as the top of the widget hierarchy
```

```
_baseWidget = _bbHistogram = XtVaCreateWidget ( _name,
                                                xmBulletinBoardWidgetClass,
                                                parent,
                                                XmNresizePolicy, XmRESIZE_GROW,
                                                (XtPointer) NULL );
```

```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component
```

```
// All variables are data members of this class
```

```
_optionMenuLHist = new VkOptionMenu ( _baseWidget, "optionMenuLHist");
_optionLUpdate = _optionMenuLHist->addAction ( "optionLUpdate",
                                                &BbHistogramUI::doOptionUpdateCallk
                                                (XtPointer) this );
```

```
_optionLCoarse = _optionMenuLHist->addAction ( "optionLCoarse",
                                                &BbHistogramUI::doOptionCoarseCallk
                                                (XtPointer) this );
```

```
_optionLFine = _optionMenuLHist->addAction ( "optionLFine",
                                                &BbHistogramUI::doOptionLFineCallback
                                                (XtPointer) this );
```

```
_optionLMapping = _optionMenuLHist->addAction ( "optionLMapping",
                                                &BbHistogramUI::doOptionMappingCal
                                                (XtPointer) this );
```

```
_dialCenter = XtVaCreateManagedWidget ( "dialCenter",
                                          sgDialWidgetClass,
                                          _baseWidget,
                                          XmNx, 438,
                                          XmNy, 92,
                                          XmNwidth, 70,
                                          XmNheight, 60,
                                          (XtPointer) NULL );
```

```
XtAddCallback ( _dialCenter,
                XmNdragCallback,
                &BbHistogramUI::centerDragCallback,
                (XtPointer) this );
```

```
XtAddCallback ( _dialCenter,
                XmNvalueChangedCallback,
                &BbHistogramUI::lowChgCallback,
                (XtPointer) this );
```

```
_dialWidth = XtVaCreateManagedWidget ( "dialWidth",
                                          sgDialWidgetClass,
                                          _baseWidget,
                                          SgNdialVisual, SgKNOB,
                                          XmNx, 512,
                                          XmNy, 92,
                                          XmNwidth, 70,
                                          XmNheight, 60,
                                          (XtPointer) NULL );
```

```
XtAddCallback ( _dialWidth,
```

[illegible]

```

_labelLHistoMin = XtVaCreateManagedWidget ( "labelLHistoMin",
                                             xmLabelWidgetClass,
                                             _baseWidget,
                                             XmNlabelType, XmSTRING,
                                             XmNx, 10,
                                             XmNy, 10,
                                             XmNwidth, 20,
                                             XmNheight, 20,
                                             (XtPointer) NULL );

XtVaSetValues ( _optionMenuLHist->baseWidget(),
                XmNx, 462,
                XmNy, 13,
                XmNwidth, 122,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuLHistogram->baseWidget(),
                XmNx, 432,
                XmNy, 50,
                XmNwidth, 152,
                XmNheight, 32,
                (XtPointer) NULL );

//---- Start editable code block: BbHistogramUI create

//---- End editable code block: BbHistogramUI create
}

const char * BbHistogramUI::className()
{
    return ("BbHistogramUI");
} // End className()

////////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////////

void BbHistogramUI::abdomenCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->abdomen ( w, callData );
}

void BbHistogramUI::boneCallback ( Widget      w,
                                  XtPointer clientData,
                                  XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->bone ( w, callData );
}

void BbHistogramUI::centerDragCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;

```



```

    obj->centerDrag ( w, callData );
}

void BbHistogramUI::doOptionCoarseCallback ( Widget      w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->doOptionCoarse ( w, callData );
}

void BbHistogramUI::doOptionLFineCallback ( Widget      w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->doOptionLFine ( w, callData );
}

void BbHistogramUI::doOptionMappingCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->doOptionMapping ( w, callData );
}

void BbHistogramUI::doOptionUpdateCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->doOptionUpdate ( w, callData );
}

void BbHistogramUI::headCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->head ( w, callData );
}

void BbHistogramUI::highChgCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->highChg ( w, callData );
}

void BbHistogramUI::lowChgCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->lowChg ( w, callData );
}

void BbHistogramUI::lungCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->lung ( w, callData );
}

```

```

void BbHistogramUI::mediastinumCallback ( Widget    w,
                                           XtPointer clientData,
                                           XtPointer callData )

{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->mediastinum ( w, callData );
}

void BbHistogramUI::spineCallback ( Widget    w,
                                    XtPointer clientData,
                                    XtPointer callData )

{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->spine ( w, callData );
}

void BbHistogramUI::widthDragCallback ( Widget    w,
                                        XtPointer clientData,
                                        XtPointer callData )

{
    BbHistogramUI* obj = ( BbHistogramUI * ) clientData;
    obj->widthDrag ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbHistogramUI::abdomen ( Widget, XtPointer )
{
    // This virtual function is called from abdomenCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::bone ( Widget, XtPointer )
{
    // This virtual function is called from boneCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::centerDrag ( Widget, XtPointer )
{
    // This virtual function is called from centerDragCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::doOptionCoarse ( Widget, XtPointer )
{
    // This virtual function is called from doOptionCoarseCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::doOptionLFine ( Widget, XtPointer )
{
    // This virtual function is called from doOptionLFineCallback.
    // This function is normally overridden by a derived class.
}

```

```
void BbHistogramUI::doOptionMapping ( Widget, XtPointer )
{
    // This virtual function is called from doOptionMappingCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::doOptionUpdate ( Widget, XtPointer )
{
    // This virtual function is called from doOptionUpdateCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::head ( Widget, XtPointer )
{
    // This virtual function is called from headCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::highChg ( Widget, XtPointer )
{
    // This virtual function is called from highChgCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::lowChg ( Widget, XtPointer )
{
    // This virtual function is called from lowChgCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::lung ( Widget, XtPointer )
{
    // This virtual function is called from lungCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::mediastinum ( Widget, XtPointer )
{
    // This virtual function is called from mediastinumCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::spine ( Widget, XtPointer )
{
    // This virtual function is called from spineCallback.
    // This function is normally overridden by a derived class.
}

void BbHistogramUI::widthDrag ( Widget, XtPointer )
{
    // This virtual function is called from widthDragCallback.
    // This function is normally overridden by a derived class.
}
```



//---- Start editable code block: End of generated code

381

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbLConfig
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLConfigUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbLConfig.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbLConfigUI and are
// available as protected data members inherited by this class
//
// XmList                _scrolledListAnatomy
// XmList                _scrolledListPatients
// XmPushButton          _buttonAccept
//
////////////////////////////////////

///----- Start editable code block: headers and declarations

#include "BbUI.h"
#include "BbVisual.h"

///----- End editable code block: headers and declarations

///----- BbLConfig Constructor

BbLConfig::BbLConfig(const char *name, Widget parent) :
    BbLConfigUI(name, parent)
{
    // This constructor calls BbLConfigUI(parent, name)
    // which calls BbLConfigUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    ///----- Start editable code block: BbLConfig constructor

```

```

init0();

//---- End editable code block: BbLConfig constructor

}    // End Constructor

BbLConfig::BbLConfig(const char *name) :
    BbLConfigUI(name)
{
    // This constructor calls BbLConfigUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbLConfig constructor 2

    init0();

    //---- End editable code block: BbLConfig constructor 2

}    // End Constructor

BbLConfig::~BbLConfig()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbLConfig destructor

    //---- End editable code block: BbLConfig destructor

}    // End Destructor

const char * BbLConfig::className() // classname
{
    return ("BbLConfig");
} // End className()

void BbLConfig::anatomy ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLConfig anatomy

    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbLConfig::anatomy is implemented:

    //::VkUnimplemented ( w, "BbLConfig::anatomy" );

    for(int i=0; i<((Patients *) (_objMag -> _patients)) -> studies -> num; i++)
        if( XmListPosSelected(w, i) ) break;

```

```

printf(" Studies:   %d\n", i);
if(i==0) i = ((Patients *) (_objMag -> _patients)) -> studies -> num - 1;
else --i;

_study_no = i;

//---- End editable code block: BbLConfig anatomy
} // End BbLConfig::anatomy()

void BbLConfig::doButtonAccept ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLConfig doButtonAccept
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbLConfig::doButtonAccept is implemented:
    //::VkUnimplemented ( w, "BbLConfig::doButtonAccept" );

    set_patients();
    ((BbUI *) (_objMag -> _bb)) -> init_patient();

    //---- End editable code block: BbLConfig doButtonAccept
} // End BbLConfig::doButtonAccept()

void BbLConfig::patients ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLConfig patients
    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;
    //--- Comment out the following line when BbLConfig::patients is implemented:
    //::VkUnimplemented ( w, "BbLConfig::patients" );

    for(int i=0; i<((Patients *) (_objMag -> _patients)) -> num; i++)
        if( XmListPosSelected(w, i) ) break;

    printf(" Patients:   %d\n", i);
    if(i==0) i = ((Patients *) (_objMag -> _patients)) -> num - 1;
    else --i;

    _patient_no = i;

    set_studies(i);

    //---- End editable code block: BbLConfig patients
} // End BbLConfig::patients()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

```

```
VkComponent *BbLConfig::CreateBbLConfig( const char *name, VkComponent *parent ) 385
```

```
{  
    VkComponent *obj = new BbLConfig ( name, parent );  
    return ( obj );  
} // End CreateBbLConfig
```

```
////////////////////////////////////  
// Function for accessing a description of the dynamic interface  
// to this class.  
////////////////////////////////////
```

```
// WARNING: This structure is different than that used with 1.1 RapidApp.  
// See the RapidApp release notes for details
```

```
struct InterfaceMap {  
    char *resourceName;  
    char *methodName;  
    char *argType;  
    char *definingClass; // Optional, if not this class  
    void (VkCallbackObject::*method)(...); // Reserved, do not set  
};
```

```
void *BbLConfig::RegisterBbLConfigInterface()  
{  
    // This structure registers information about this class  
    // that allows RapidApp to create and manipulate an instance.  
    // Each entry provides a resource name that will appear in the  
    // resource manager palette when an instance of this class is  
    // selected, the name of the member function as a string,  
    // the type of the single argument to this function, and an  
    // optional argument indicating the class that defines this function.  
    // All member functions must have the form  
    //  
    //     void memberFunction ( Type );  
    //  
    // where "Type" is one of:  
    //     const char *      (Use XmRString)  
    //     Boolean           (Use XmRBoolean)  
    //     int                (Use XmRInt)  
    //     float             (Use XmRFloat)  
    //     No argument       (Use VkRNoArg or "NoArg")  
    //     A filename        (Use VkRFilename or "Filename")  
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")  
    //     A callback        (Use XmRCallback)
```

```
static InterfaceMap map[] = {  
    //---- Start editable code block: BbLConfigUI resource table  
  
    // { "resourceName", "setAttribute", XmRString},  
    //---- End editable code block: BbLConfigUI resource table  
    { NULL }, // MUST be NULL terminated  
};
```

```
return map;  
} // End RegisterBbLConfigInterface()
```


//---- Start editable code block: End of generated code

void BbLConfig::init0()

```
{
    _patient_no = -1;
    _study_no = -1;
}
```

Patients *BbLConfig::init()

```
{
    FILE *fp;
    int i;

    Patients *patients = new Patients;

    if( (fp = fopen("input.patients", "r")) == NULL )
        return NULL;
    fscanf(fp, "%d", &(patients->num));
    patients->patient_info = new Patient_Info[patients->num];

    for(i=0; i<patients->num; i++)
    {
        fscanf(fp, "%s", patients->patient_info[i].name);
        fscanf(fp, "%s", patients->patient_info[i].dir);
    }
    fclose(fp);

    XmString item;
    char str[300];

    XmListDeleteAllItems(_scrolledListPatients);
    for(i=0; i<patients->num; i++)
    {
        sprintf(str, "%d %s", i+1, patients->patient_info[i].name);
        item = XmStringCreateSimple(str);
        XmListAddItem(_scrolledListPatients, item, i+1);
    }

    patients->studies = NULL;

    return patients;
}
```

void BbLConfig::set_patients()

```
{
    sprintf(_objMag -> msgsLoaded.img_dir, "%s",
        ((Patients *) (_objMag -> _patients)) -> patient_info[_patient_no].dir);
    _objMag -> msgsLoaded.img_exam =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].exam;
    _objMag -> msgsLoaded.img_series =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].series;
    _objMag -> msgsLoaded.img_start =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].start;
    _objMag -> msgsLoaded.img_end =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].end;
    _objMag -> msgsLoaded.img_start2 =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].start2;
    _objMag -> msgsLoaded.img_end2 =
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].end2;
    sprintf(_objMag -> msgsLoaded.img_type, "%s",
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].type);
    sprintf(_objMag -> msgsLoaded.img_anatomy, "%s",
        ((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].anatomy);
    sprintf(_objMag -> msgsLoaded.img_ref, "%s",
```

```
((Patients *) (_objMag -> _patients)) -> studies -> study_info[_study_no].ref);
```

387

```
void BbLConfig::set_studies(int index_study)
```

```
{
    FILE *fp;
    int i;
    char str[300], str1[100], str2[100], str3[100];

    Studies *studies = new Studies;

    sprintf(str, "%s%/input.studies", ((Patients *) (_objMag -> _patients)) -> patient_i);
    printf(" %s \n", str);

    if( (fp = fopen(str, "r")) == NULL )
        return;

    fscanf(fp, "%d", &(studies->num));
    studies->study_info = new Study_Info[studies->num];

    printf(" %d \n", studies->num);

    for(i=0; i<studies->num; i++)
    {
        fscanf(fp, "%s", studies->study_info[i].name);
        fscanf(fp, "%s", studies->study_info[i].type);
        fscanf(fp, "%s", studies->study_info[i].anatomy);
        fscanf(fp, "%d", &(studies->study_info[i].exam));
        fscanf(fp, "%d", &(studies->study_info[i].series));
        fscanf(fp, "%d", &(studies->study_info[i].start));
        fscanf(fp, "%d", &(studies->study_info[i].end));
        fscanf(fp, "%d", &(studies->study_info[i].start2));
        fscanf(fp, "%d", &(studies->study_info[i].end2));
        fscanf(fp, "%s\n", studies->study_info[i].ref);

        printf(" %s \n", studies->study_info[i].name);
    }

    fgets(str1, 100, fp);
    fgets(str2, 100, fp);
    fgets(str3, 100, fp);
    ((BbVisual *) (_objMag -> _RVis1)) -> set_info(str1, str2, str3);

    fclose(fp);

    XmString item;

    XmListDeleteAllItems(_scrolledListAnatomy);
    for(i=0; i<studies->num; i++)
    {
        sprintf(str, "%d %s", i+1, studies->study_info[i].name);
        item = XmStringCreateSimple(str);
        XmListAddItem(_scrolledListAnatomy, item, 0);
        printf(" %s\n", str);
    }

    if(_objMag -> _patients -> studies != NULL)
    {
        delete _objMag -> _patients -> studies -> study_info;
        delete _objMag -> _patients -> studies;
    }

    _objMag -> _patients -> studies = studies;
}
```

}

//---- End editable code block: End of generated code

388

```

////////////////////////////////////
//
// Source file for BbLConfigUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
-//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbLConfigUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
/*--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbLConfigUI::_defaultBbLConfigUIResources[] = {
    "buttonAccept.labelString: Accept",
    "tabLabel: Config",

    //--- Start editable code block: BbLConfigUI Default Resources

    //--- End editable code block: BbLConfigUI Default Resources

    (char*)NULL
};

BbLConfigUI::BbLConfigUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //--- Start editable code block: BbLConfig constructor 2

    //--- End editable code block: BbLConfig constructor 2

```

```
} // End Constructor
```

```
BbLConfigUI::BbLConfigUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //----- Start editable code block: BbLConfig pre-create

    //----- End editable code block: BbLConfig pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //----- Start editable code block: BbLConfig constructor

    //----- End editable code block: BbLConfig constructor

} // End Constructor
```

```
BbLConfigUI::~BbLConfigUI()
{
    // Base class destroys widgets

    //----- Start editable code block: BbLConfigUI destructor

    //----- End editable code block: BbLConfigUI destructor
} // End destructor
```

```
void BbLConfigUI::create ( Widget parent )
{
    Arg      args[6];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBbLConfigUIResources );

    // Create an unmanaged widget as the top of the widget hierarchy
    _baseWidget = _bbLConfig = XtVaCreateWidget ( _name,
                                                    xmBulletinBoardWidgetClass,
                                                    parent,
                                                    XmNresizePolicy, XmRESIZE_GROW,
                                                    (XtPointer) NULL );

    // install a callback to guard against unexpected widget destruction
    installDestroyHandler();

    // Create widgets used in this component
```

// All variables are data members of this class

391

```
_scrolledWindowAnatomy = XtVaCreateManagedWidget ( "scrolledWindowAnatomy",  
                                                    xmScrolledWindowWidgetClass,  
                                                    _baseWidget,  
                                                    XmNscrollBarDisplayPolicy, XmS  
                                                    XmNx, 226,  
                                                    XmNy, 10,  
                                                    XmNwidth, 290,  
                                                    XmNheight, 140,  
                                                    (XtPointer) NULL );
```

```
_scrolledListAnatomy = XtVaCreateManagedWidget ( "scrolledListAnatomy",  
                                                  xmListWidgetClass,  
                                                  _scrolledWindowAnatomy,  
                                                  XmNlistSizePolicy, XmCONSTANT,  
                                                  XmNwidth, 284,  
                                                  XmNheight, 134,  
                                                  (XtPointer) NULL );
```

```
XtAddCallback ( _scrolledListAnatomy,  
               XmNbrowseSelectionCallback,  
               &BbLConfigUI::anatomyCallback,  
               (XtPointer) this );
```

```
_scrolledWindowPatients = XtVaCreateManagedWidget ( "scrolledWindowPatients",  
                                                     xmScrolledWindowWidgetClass,  
                                                     _baseWidget,  
                                                     XmNscrollBarDisplayPolicy, Xm  
                                                     XmNx, 10,  
                                                     XmNy, 10,  
                                                     XmNwidth, 200,  
                                                     XmNheight, 140,  
                                                     (XtPointer) NULL );
```

```
_scrolledListPatients = XtVaCreateManagedWidget ( "scrolledListPatients",  
                                                  xmListWidgetClass,  
                                                  _scrolledWindowPatients,  
                                                  XmNlistSizePolicy, XmCONSTANT,  
                                                  XmNwidth, 194,  
                                                  XmNheight, 134,  
                                                  (XtPointer) NULL );
```

```
XtAddCallback ( _scrolledListPatients,  
               XmNbrowseSelectionCallback,  
               &BbLConfigUI::patientsCallback,  
               (XtPointer) this );
```

```
_buttonAccept = XtVaCreateManagedWidget ( "buttonAccept",  
                                            xmPushButtonWidgetClass,  
                                            _baseWidget,  
                                            XmNlabelType, XmSTRING,  
                                            XmNx, 532,  
                                            XmNy, 88,  
                                            XmNwidth, 60,  
                                            XmNheight, 60,  
                                            (XtPointer) NULL );
```

```
XtAddCallback ( _buttonAccept,  
               XmNactivateCallback,  
               &BbLConfigUI::doButtonAcceptCallback,  
               (XtPointer) this );
```

```

//---- Start editable code block: BbLConfigUI create

//---- End editable code block: BbLConfigUI create
}

const char * BbLConfigUI::className()
{
    return ("BbLConfigUI");
} // End className()

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

void BbLConfigUI::anatomyCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbLConfigUI* obj = ( BbLConfigUI * ) clientData;
    obj->anatomy ( w, callData );
}

void BbLConfigUI::doButtonAcceptCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLConfigUI* obj = ( BbLConfigUI * ) clientData;
    obj->doButtonAccept ( w, callData );
}

void BbLConfigUI::patientsCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbLConfigUI* obj = ( BbLConfigUI * ) clientData;
    obj->patients ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbLConfigUI::anatomy ( Widget, XtPointer )
{
    // This virtual function is called from anatomyCallback.
    // This function is normally overridden by a derived class.
}

void BbLConfigUI::doButtonAccept ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptCallback.
    // This function is normally overridden by a derived class.
}

```

```
void BbLConfigUI::patient(Widget, XtPointer )
```

393

```
{  
    // This virtual function is called from patientsCallback.  
    // This function is normally overridden by a derived class.  
}
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```



```

////////////////////////////////////
//
// Source file for BbLPCMRA
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLPCMRAUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbLPCMRA.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbLPCMRAUI and are
// available as protected data members inherited by this class
//
// XmPushButton          _buttonHideLocalizer
// XmPushButton          _buttonShowLocalizer
//
////////////////////////////////////

///---- Start editable code block: headers and declarations

#include "Utility.h"

///---- End editable code block: headers and declarations

///---- BbLPCMRA Constructor

BbLPCMRA::BbLPCMRA(const char *name, Widget parent) :
    BbLPCMRAUI(name, parent)
{
    // This constructor calls BbLPCMRAUI(parent, name)
    // which calls BbLPCMRAUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    ///---- Start editable code block: BbLPCMRA constructor

    ///---- End editable code block: BbLPCMRA constructor

```

```
} // End Constructor
```

```
BbLPCMRA::BbLPCMRA(const char *name) :  
    BbLPCMRAUI(name)
```

```
{  
    // This constructor calls BbLPCMRAUI(name)  
    // which does not create any widgets. Usually, this  
    // constructor is not used  
  
    //----- Start editable code block: BbLPCMRA constructor 2  
  
    //----- End editable code block: BbLPCMRA constructor 2
```

```
} // End Constructor
```

```
BbLPCMRA::~BbLPCMRA()
```

```
{  
    // The base class destructors are responsible for  
    // destroying all widgets and objects used in this component.  
    // Only additional items created directly in this class  
    // need to be freed here.  
  
    //----- Start editable code block: BbLPCMRA destructor  
  
    //----- End editable code block: BbLPCMRA destructor
```

```
} // End Destructor
```

```
const char * BbLPCMRA::className() // classname  
{  
    return ("BbLPCMRA");  
} // End className()
```

```
void BbLPCMRA::doButtonHideLocalizer ( Widget w, XtPointer callData )  
{  
    //----- Start editable code block: BbLPCMRA doButtonHideLocalizer  
  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
  
    //--- Comment out the following line when BbLPCMRA::doButtonHideLocalizer is implem  
  
    //::VkUnimplemented ( w, "BbLPCMRA::doButtonHideLocalizer" );  
  
    if(_objMag -> msgsRight.img_select == RIGHT_IMG_REF  
        && _objMag -> msgsRight.img_space == IMAGE_2D)  
        _objMag -> _imgView2 -> display();  
  
    //----- End editable code block: BbLPCMRA doButtonHideLocalizer  
  
} // End BbLPCMRA::doButtonHideLocalizer()
```

```
void BbLPCMRA::doButtonShowLocalizer ( Widget w, XtPointer callData )
```

396

```
{
    //---- Start editable code block: BbLPCMRA doButtonShowLocalizer

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLPCMRA::doButtonShowLocalizer is implem
    //::VkUnimplemented ( w, "BbLPCMRA::doButtonShowLocalizer" );

    if(_objMag -> msgsRight.img_select == RIGHT_IMG_REF
        && _objMag -> msgsRight.img_space == IMAGE_2D)
        _objMag -> localizer();

    //---- End editable code block: BbLPCMRA doButtonShowLocalizer
} // End BbLPCMRA::doButtonShowLocalizer()
```

```
////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////
```

```
VkComponent *BbLPCMRA::CreateBbLPCMRA( const char *name, Widget parent )
{
    VkComponent *obj = new BbLPCMRA ( name, parent );
    return ( obj );
} // End CreateBbLPCMRA
```

```
////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////
```

```
// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details
```

```
struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};
```

```
void *BbLPCMRA::RegisterBbLPCMRAInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
```

```

//      void memberFunction ( Type );
//
// where "Type" is one of:
//      const char *      (Use XmRString)
//      Boolean           (Use XmRBoolean)
//      int               (Use XmRInt)
//      float            (Use XmRFloat)
//      No argument       (Use VkrNoArg or "NoArg")
//      A filename        (Use VkrFilename or "Filename")
//      An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
//      A callback        (Use XmRCallback)

static InterfaceMap map[] = {
//---- Start editable code block: BbLPCMRAUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbLPCMRAUI resource table
    { NULL }, // MUST be NULL terminated
};

return map;
} // End RegisterBbLPCMRAInterface()

//---- End of generated code

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbLPCMRAUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbLPCMRAUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbLPCMRAUI::_defaultBbLPCMRAUIResources[] = {
    "**buttonHideLocalizer.labelString: Hide Localizer",
    "**buttonShowLocalizer.labelString: Show Localizer",
    "**tabLabel: Localizer",

    //---- Start editable code block: BbLPCMRAUI Default Resources

    //---- End editable code block: BbLPCMRAUI Default Resources

    (char*)NULL
};

BbLPCMRAUI::BbLPCMRAUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbLPCMRA constructor 2

    //---- End editable code block: BbLPCMRA constructor 2

```

```
BbLPCMRAUI::BbLPCMRAUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //----- Start editable code block: BbLPCMRA pre-create

    //----- End editable code block: BbLPCMRA pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //----- Start editable code block: BbLPCMRA constructor

    //----- End editable code block: BbLPCMRA constructor

} // End Constructor
```

```
BbLPCMRAUI::~BbLPCMRAUI()
{
    // Base class destroys widgets

    //----- Start editable code block: BbLPCMRAUI destructor

    //----- End editable code block: BbLPCMRAUI destructor
} // End destructor
```

```
void BbLPCMRAUI::create ( Widget parent )
{
    Arg      args[6];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBbLPCMRAUIResources );

    // Create an unmanaged widget as the top of the widget hierarchy
    _baseWidget = _bbLPCMRA = XtVaCreateWidget ( _name,
                                                xmBulletinBoardWidgetClass,
                                                parent,
                                                XmNresizePolicy, XmRESIZE_GROW,
                                                (XtPointer) NULL );

    // install a callback to guard against unexpected widget destruction
    installDestroyHandler();

    // Create widgets used in this component
    // All variables are data members of this class
```

```

_buttonHideLocalizer XtVaCreateManagedWidget ( "buttonHideLocalizer", 400
                                                xmPushButtonWidgetClass,
                                                _baseWidget,
                                                XmNlabelType, XmSTRING,
                                                XmNx, 300,
                                                XmNy, 60,
                                                XmNwidth, 120,
                                                XmNheight, 30,
                                                (XtPointer) NULL );

```

```

XtAddCallback ( _buttonHideLocalizer,
                XmNactivateCallback,
                &BbLPCMRAUI::doButtonHideLocalizerCallback,
                (XtPointer) this );

```

```

_buttonShowLocalizer = XtVaCreateManagedWidget ( "buttonShowLocalizer",
                                                  xmPushButtonWidgetClass,
                                                  _baseWidget,
                                                  XmNlabelType, XmSTRING,
                                                  XmNx, 150,
                                                  XmNy, 60,
                                                  XmNwidth, 116,
                                                  XmNheight, 30,
                                                  (XtPointer) NULL );

```

```

XtAddCallback ( _buttonShowLocalizer,
                XmNactivateCallback,
                &BbLPCMRAUI::doButtonShowLocalizerCallback,
                (XtPointer) this );

```

```

//---- Start editable code block: BbLPCMRAUI create

```

```

//---- End editable code block: BbLPCMRAUI create

```

```

}

const char * BbLPCMRAUI::className()
{
    return ("BbLPCMRAUI");
}
    // End className()

```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void BbLPCMRAUI::doButtonHideLocalizerCallback ( Widget    w,
                                                  XtPointer clientData,
                                                  XtPointer callData )

{
    BbLPCMRAUI* obj = ( BbLPCMRAUI * ) clientData;
    obj->doButtonHideLocalizer ( w, callData );
}

```

```

void BbLPCMRAUI::doButtonShowLocalizerCallback ( Widget    w,
                                                  XtPointer clientData,
                                                  XtPointer callData )

{
    BbLPCMRAUI* obj = ( BbLPCMRAUI * ) clientData;

```

```
obj->doButtonShowLocalizer ( w, callData );
```

401

```
////////////////////////////////////  
// The following functions are called from the menu items  
// in this window.  
////////////////////////////////////
```

```
void BbLPCMRAUI::doButtonHideLocalizer ( Widget, XtPointer )  
{  
    // This virtual function is called from doButtonHideLocalizerCallback.  
    // This function is normally overridden by a derived class.  
}
```

```
void BbLPCMRAUI::doButtonShowLocalizer ( Widget, XtPointer )  
{  
    // This virtual function is called from doButtonShowLocalizerCallback.  
    // This function is normally overridden by a derived class.  
}
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```


User: meide
Host: phoenix
Class: phoenix
Job: BbHistogram.C

402

```

////////////////////////////////////
//
// Source file for BbLROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbLROI.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbLROIUI and are
// available as protected data members inherited by this class
//
// XmPushButton                _buttonAcceptArea
// VkOptionMenu *              _optionMenu6
// VkMenuItem *                _optionDraw
// VkMenuItem *                _optionModify
// VkMenuItem *                _optionEraseLeft
// VkMenuItem *                _optionEraseRight
// VkOptionMenu *              _optionMenuColor
// VkMenuItem *                _optionRed
// VkMenuItem *                _optionGreen
// VkMenuItem *                _optionBlue
// VkMenuItem *                _optionYellow
// VkMenuItem *                _optionWhite
// VkMenuItem *                _optionBlack
// VkOptionMenu *              _optionMenuROIType
// VkMenuItem *                _optionRectangle
// VkMenuItem *                _optionFreeHand
// VkMenuItem *                _optionPolygon
// VkMenuItem *                _optionEllipse
// XmPushButton                _buttonHide
// XmPushButton                _buttonShow
//
////////////////////////////////////

///---- Start editable code block: headers and declarations

```

```

#include "ROI.h"
#include "Utility.h"
#include "DeckRTabbedDeck.h"
#include "BbFlow.h"
#include <Vk/VkDeck.h>

//---- End editable code block: headers and declarations

//---- BbLROI Constructor

BbLROI::BbLROI(const char *name, Widget parent) :
    BbLROIUI(name, parent)
{
    // This constructor calls BbLROIUI(parent, name)
    // which calls BbLROIUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbLROI constructor
    init();

    //---- End editable code block: BbLROI constructor

}    // End Constructor

BbLROI::BbLROI(const char *name) :
    BbLROIUI(name)
{
    // This constructor calls BbLROIUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbLROI constructor 2
    init();

    //---- End editable code block: BbLROI constructor 2

}    // End Constructor

BbLROI::~BbLROI()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbLROI destructor

    //---- End editable code block: BbLROI destructor

}    // End Destructor

```

```
{
    return ("BbLROI");
} // End className()
```

```
void BbLROI::doButtonAccept ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doButtonAccept

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doButtonAccept is implemented:

    //::VkUnimplemented ( w, "BbLROI::doButtonAccept" );
    if(_objMag -> _imgView2 -> _ROI != NULL)
    {
        if(_objMag->_imgView2->_ROI->_area == NULL)
            _objMag->_imgView2->_ROI->set_area();
        if(_objMag->_imgView2->_ROI->_area != NULL)
            _objMag->_imgView2->_ROI->set_areaOrg(_objMag->_imgView2->_zoom);
    }

    //----- End editable code block: BbLROI doButtonAccept
} // End BbLROI::doButtonAccept()
```

```
void BbLROI::doButtonHide ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doButtonHide

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doButtonHide is implemented:

    //::VkUnimplemented ( w, "BbLROI::doButtonHide" );
    if(_objMag -> msgsRight.img_space == IMAGE_2D)
    {
        //if(_roi_mode == ROI_LEFT)
            _objMag-> _imgView -> HideROI();
        //else if(_roi_mode == ROI_RIGHT)
            _objMag-> _imgView2 -> HideROI();
    }

    //----- End editable code block: BbLROI doButtonHide
} // End BbLROI::doButtonHide()
```

```
void BbLROI::doButtonShow ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doButtonShow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doButtonShow is implemented:

    //::VkUnimplemented ( w, "BbLROI::doButtonShow" );
    if(_objMag -> msgsRight.img_space == IMAGE_2D)
    {
        //if(_roi_mode == ROI_LEFT)
            _objMag-> _imgView -> ShowROI();
        //else if(_roi_mode == ROI_RIGHT)
            _objMag-> _imgView2 -> ShowROI();
    }
}
```

```

}

//----- End editable code block: BbLROI doButtonShow

} // End BbLROI::doButtonShow()

void BbLROI::doOptionBlack ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doOptionBlack

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionBlack is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionBlack" );

    _roi_color = COLOR_BLACK;
    set_color();

    //----- End editable code block: BbLROI doOptionBlack
} // End BbLROI::doOptionBlack()

void BbLROI::doOptionBlue ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doOptionBlue

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionBlue is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionBlue" );

    _roi_color = COLOR_BLUE;
    set_color();

    //----- End editable code block: BbLROI doOptionBlue
} // End BbLROI::doOptionBlue()

void BbLROI::doOptionDraw ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doOptionDraw

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionDraw is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionDraw" );

    _roi_action = ROI_REDEFINE;
    changeROI();

    //----- End editable code block: BbLROI doOptionDraw
} // End BbLROI::doOptionDraw()

void BbLROI::doOptionEllipse ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLROI doOptionEllipse

```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData407

//--- Comment out the following line when BbLROI::doOptionEllipse is implemented:

//::VkUnimplemented ( w, "BbLROI::doOptionEllipse" );

    _roi_type = ROI_ELLIPSE;
    _roi_action = ROI_REDEFINE;
    _optionMenu6 -> set("optionDraw");
    changeROI();

//---- End editable code block: BbLROI doOptionEllipse
}    // End BbLROI::doOptionEllipse()

void BbLROI::doOptionEraseLeft ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionEraseLeft

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionEraseLeft is implemented:

    //::VkUnimplemented ( w, "BbLROI::doOptionEraseLeft" );

    if(_objMag -> msgsLeft.img_space == IMAGE_2D)
        _objMag-> _imgView -> EraseROI();

    //---- End editable code block: BbLROI doOptionEraseLeft
}    // End BbLROI::doOptionEraseLeft()

void BbLROI::doOptionEraseRight ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionEraseRight

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionEraseRight is implemented:

    //::VkUnimplemented ( w, "BbLROI::doOptionEraseRight" );

    if(_objMag -> msgsRight.img_space == IMAGE_2D)
        _objMag-> _imgView2 -> EraseROI();

    //---- End editable code block: BbLROI doOptionEraseRight
}    // End BbLROI::doOptionEraseRight()

void BbLROI::doOptionFreeHand ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionFreeHand

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLROI::doOptionFreeHand is implemented:

    //::VkUnimplemented ( w, "BbLROI::doOptionFreeHand" );

```

```

_roi_type = ROI_FREEHAND;
_roi_action = ROI_REDEFINE;
_optionMenu6 -> set("optionDraw");
changeROI();

//---- End editable code block: BbLROI doOptionFreeHand
} // End BbLROI::doOptionFreeHand()

void BbLROI::doOptionGreen ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionGreen
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbLROI::doOptionGreen is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionGreen" );
    _roi_color = COLOR_GREEN;
    set_color();

    //---- End editable code block: BbLROI doOptionGreen
} // End BbLROI::doOptionGreen()

void BbLROI::doOptionModify ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionModify
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbLROI::doOptionModify is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionModify" );
    _roi_action = ROI_MODIFY;
    changeROI();

    //---- End editable code block: BbLROI doOptionModify
} // End BbLROI::doOptionModify()

void BbLROI::doOptionPolygon ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLROI doOptionPolygon
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbLROI::doOptionPolygon is implemented:
    //::VkUnimplemented ( w, "BbLROI::doOptionPolygon" );
    _roi_type = ROI_POLYGON;
    _roi_action = ROI_REDEFINE;
    _optionMenu6 -> set("optionDraw");
    changeROI();

    //---- End editable code block: BbLROI doOptionPolygon
}

```

```
void BbLROI::doOptionRectangle ( Widget w, XtPointer callData )
{
```

```
    //----- Start editable code block: BbLROI doOptionRectangle
```

```
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
    //--- Comment out the following line when BbLROI::doOptionRectangle is implemented:
```

```
    //::VkUnimplemented ( w, "BbLROI::doOptionRectangle" );
```

```
        _roi_type = ROI_RECTANGLE;
```

```
        _roi_action = ROI_REDEFINE;
```

```
        _optionMenu6 -> set("optionDraw");
```

```
        changeROI();
```

```
    //----- End editable code block: BbLROI doOptionRectangle
```

```
} // End BbLROI::doOptionRectangle()
```

```
void BbLROI::doOptionRed ( Widget w, XtPointer callData )
{
```

```
    //----- Start editable code block: BbLROI doOptionRed
```

```
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
    //--- Comment out the following line when BbLROI::doOptionRed is implemented:
```

```
    //::VkUnimplemented ( w, "BbLROI::doOptionRed" );
```

```
    _roi_color = COLOR_RED;
```

```
    set_color();
```

```
    //----- End editable code block: BbLROI doOptionRed
```

```
} // End BbLROI::doOptionRed()
```

```
void BbLROI::doOptionWhite ( Widget w, XtPointer callData )
{
```

```
    //----- Start editable code block: BbLROI doOptionWhite
```

```
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
    //--- Comment out the following line when BbLROI::doOptionWhite is implemented:
```

```
    //::VkUnimplemented ( w, "BbLROI::doOptionWhite" );
```

```
    _roi_color = COLOR_WHITE;
```

```
    set_color();
```

```
    //----- End editable code block: BbLROI doOptionWhite
```

```
} // End BbLROI::doOptionWhite()
```

```
void BbLROI::doOptionYellow ( Widget w, XtPointer callData )
{
```

```
    //----- Start editable code block: BbLROI doOptionYellow
```

```
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```



```

//--- Comment out the following line when BbLROI::doOptionYellow is implemented:
//::VkUnimplemented ( w, "BbLROI::doOptionYellow" );
_roi_color = COLOR_YELLOW;
set_color();

//---- End editable code block: BbLROI doOptionYellow
} // End BbLROI::doOptionYellow()

```

```

/////////////////////////////////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
/////////////////////////////////////////////////////////////////

```

```

VkComponent *BbLROI::CreateBbLROI( const char *name, Widget parent )
{
    VkComponent *obj = new BbLROI ( name, parent );
    return ( obj );
} // End CreateBbLROI

```

```

/////////////////////////////////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
/////////////////////////////////////////////////////////////////

```

```

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbLROI::RegisterBbLROIInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int                (Use XmRInt)
    //     float              (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg"

```

```

//      A filename      (Use VkRFilename or "Filename")
//      An enumeration   Use "Enumeration:ClassName:Type" VALUE1, VALUE2, VALUE3")
//      A callback       (Use XmRCallback)

```

```

static InterfaceMap map[] = {
//---- Start editable code block: BbLROIUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbLROIUI resource table
    { NULL }, // MUST be NULL terminated
};

```

```

    return map;
} // End RegisterBbLROIInterface()

```

```

//---- End of generated code

```

```

//---- Start editable code block: End of generated code

```

```

void BbLROI::init()

```

```

{
    _roi_mode = ROI_LEFT;
    _roi_type = ROI_RECTANGLE;
    _roi_action = ROI_REDEFINE;
    _roi_color = COLOR_RED;
}

```

```

void BbLROI::init2()

```

```

{
    //XmToggleButtonSetState(_toggleLeft1, TRUE, TRUE);
    //XmToggleButtonSetState(_toggleRedefine, TRUE, TRUE);
    //XmToggleButtonSetState(_toggleROIRectangle, TRUE, TRUE);
}

```

```

void BbLROI::initROI()

```

```

{
    //_objMag->msgsLeft.roi_mode == ROI_RIGHT;
    _objMag->msgsRight.roi_type = ROI_FREEHAND;
    _objMag->msgsRight.roi_action = ROI_REDEFINE;
    _objMag->_imgView2 -> _roi_type = ROI_FREEHAND;
    _objMag->_imgView2 -> _roi_action = ROI_REDEFINE;

    //XmToggleButtonSetState(_toggleROIFree, TRUE, TRUE);
    //XmToggleButtonSetState(_toggleRight1, TRUE, TRUE);
    //XmToggleButtonSetState(_toggleRedefine, TRUE, TRUE);
}

```

```

void BbLROI::changeROI()

```

```

{
    _objMag-> _imgView -> _roi_type = _roi_type;
    _objMag-> _imgView -> _roi_action = _roi_action;
    _objMag->msgsLeft.roi_type = _roi_type;
    _objMag->msgsLeft.roi_action = _roi_action;

    _objMag-> _imgView2 -> _roi_type = _roi_type;
    _objMag-> _imgView2 -> _roi_action = _roi_action;
    _objMag->msgsRight.roi_type = _roi_type;
    _objMag->msgsRight.roi_action = _roi_action;
}

```

```
set_color();
```

412

```
}
```

```
void BbLROI::set_color()
```

```
{
    //if(_roi_mode == ROI_LEFT)
        _objMag -> _imgView -> set_color(_roi_color);
    //else if(_roi_mode == ROI_RIGHT)
        _objMag -> _imgView2 -> set_color(_roi_color);
}
```

```
void BbLROI::set_type()
```

```
{
    _roi_action = ROI_REDEFINE;
    //_optionMenu6 -> set("optionDraw");

    _roi_type = _objMag-> _imgView -> _roi_type;
    if( _objMag-> _imgView -> _roi_type == ROI_RECTANGLE)
        _optionMenuROIType -> set("optionRectangle");
    else if( _objMag-> _imgView -> _roi_type == ROI_ELLIPSE)
        _optionMenuROIType -> set("optionEllipse");
    else if( _objMag-> _imgView -> _roi_type == ROI_POLYGON)
        _optionMenuROIType -> set("optionPolygon");
    else if( _objMag-> _imgView -> _roi_type == ROI_FREEHAND)
        _optionMenuROIType -> set("optionFreeHand");

    _roi_type = _objMag-> _imgView2 -> _roi_type;
    if( _objMag-> _imgView2 -> _roi_type == ROI_RECTANGLE)
        _optionMenuROIType -> set("optionRectangle");
    else if( _objMag-> _imgView2 -> _roi_type == ROI_ELLIPSE)
        _optionMenuROIType -> set("optionEllipse");
    else if( _objMag-> _imgView2 -> _roi_type == ROI_POLYGON)
        _optionMenuROIType -> set("optionPolygon");
    else if( _objMag-> _imgView2 -> _roi_type == ROI_FREEHAND)
        _optionMenuROIType -> set("optionFreeHand");

}
```

```
//----- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbLROIUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbLROIUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbLROIUI::_defaultBbLROIUIResources[] = {
    "*buttonAcceptArea.labelString: Accept",
    "*buttonHide.labelString: Hide",
    "*buttonShow.labelString: Show",
    "*optionBlack.labelString: Black",
    "*optionBlue.labelString: Blue",
    "*optionDraw.labelString: Draw",
    "*optionEllipse.labelString: Ellipse",
    "*optionEraseLeft.labelString: Erase Left",
    "*optionEraseRight.labelString: Erase Right",
    "*optionFreeHand.labelString: FreeHand",
    "*optionGreen.labelString: Green",
    "*optionMenuColor.labelString: ",
    "*optionMenuROIType.labelString: ",
    "*optionModify.labelString: Modify",
    "*optionPolygon.labelString: Polygon",
    "*optionRectangle.labelString: Rectangle",
    "*optionRed.labelString: Red",
    "*optionWhite.labelString: White",
    "*optionYellow.labelString: Yellow",
    "*tabLabel: ROI",

    //---- Start editable code block: BbLROIUI Default Resources

```

```
(char*)NULL
};

BbLROIUI::BbLROIUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbLROI constructor 2

    //---- End editable code block: BbLROI constructor 2

} // End Constructor

BbLROIUI::BbLROIUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //---- Start editable code block: BbLROI pre-create

    //---- End editable code block: BbLROI pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: BbLROI constructor

    //---- End editable code block: BbLROI constructor

} // End Constructor

BbLROIUI::~BbLROIUI()
{
    // Base class destroys widgets

    //---- Start editable code block: BbLROIUI destructor

    //---- End editable code block: BbLROIUI destructor
} // End destructor

void BbLROIUI::create ( Widget parent )
{
    Arg      args[9];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
```

```
// Create an unmanaged widget as the top of the widget hierarchy
```

[illegible]

```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component
// All variables are data members of this class
```

[illegible]

```
XtAddCallback ( _buttonAcceptArea,  
                XmNactivateCallback,  
                &BbLROtUI::doButtonAcceptCallback,  
                (XtPointer) this );
```

```
_optionMenu6 = new VkOptionMenu ( _baseWidget, "optionMenu6");  
_optionDraw = _optionMenu6->addAction ( "optionDraw",  
                                         &BbLROIUI::doOptionDrawCallback,  
                                         (XtPointer) this );
```

[illegible][illegible]

```
_optionEraseRight = _optionMenu6->addAction ( "optionEraseRight",
&BbLROIUI::doOptionEraseRightCallbac
(XtPointer) this );
```

[illegible][illegible][illegible]

```

_optionYellow = _optionMenuColor->addAction ( "optionYellow",
                                                &BbLROIUI::doOptionYellowCallback,
                                                (XtPointer) this );

_optionWhite = _optionMenuColor->addAction ( "optionWhite",
                                                &BbLROIUI::doOptionWhiteCallback,
                                                (XtPointer) this );

_optionBlack = _optionMenuColor->addAction ( "optionBlack",
                                                &BbLROIUI::doOptionBlackCallback,
                                                (XtPointer) this );

_optionMenuROIType = new VkOptionMenu ( _baseWidget, "optionMenuROIType");
_optionRectangle = _optionMenuROIType->addAction ( "optionRectangle",
                                                &BbLROIUI::doOptionRectangleCal
                                                (XtPointer) this );

_optionFreeHand = _optionMenuROIType->addAction ( "optionFreeHand",
                                                &BbLROIUI::doOptionFreeHandCallb
                                                (XtPointer) this );

_optionPolygon = _optionMenuROIType->addAction ( "optionPolygon",
                                                &BbLROIUI::doOptionPolygonCallbac
                                                (XtPointer) this );

_optionEllipse = _optionMenuROIType->addAction ( "optionEllipse",
                                                &BbLROIUI::doOptionEllipseCallbac
                                                (XtPointer) this );

_buttonHide = XtVaCreateManagedWidget ( "buttonHide",
                                         xmPushButtonWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 70,
                                         XmNy, 109,
                                         XmNwidth, 90,
                                         XmNheight, 50,
                                         (XtPointer) NULL );

XtAddCallback ( _buttonHide,
               XmNactivateCallback,
               &BbLROIUI::doButtonHideCallback,
               (XtPointer) this );

_buttonShow = XtVaCreateManagedWidget ( "buttonShow",
                                         xmPushButtonWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 70,
                                         XmNy, 59,
                                         XmNwidth, 90,
                                         XmNheight, 50,
                                         (XtPointer) NULL );

XtAddCallback ( _buttonShow,
               XmNactivateCallback,
               &BbLROIUI::doButtonShowCallback,
               (XtPointer) this );

XtVaSetValues ( _optionMenu6->baseWidget(),
               XmNx, 357,
               XmNy, 20,
               XmNwidth, 142,
               XmNheight, 32,

```

```

        (XtPointer) NULL );
XtVaSetValues ( _optionMenuColor->baseWidget(),
                XmNx, 393,
                XmNy, 110,
                XmNwidth, 106,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuROIType->baseWidget(),
                XmNx, 369,
                XmNy, 65,
                XmNwidth, 130,
                XmNheight, 32,
                (XtPointer) NULL );

//----- Start editable code block: BbLROIUI create

//----- End editable code block: BbLROIUI create
}

const char * BbLROIUI::className()
{
    return ("BbLROIUI");
} // End className()

////////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////////

void BbLROIUI::doButtonAcceptCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doButtonAccept ( w, callData );
}

void BbLROIUI::doButtonHideCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doButtonHide ( w, callData );
}

void BbLROIUI::doButtonShowCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doButtonShow ( w, callData );
}

void BbLROIUI::doOptionBlackCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionBlack ( w, callData );
}

void BbLROIUI::doOptionBlueCallback ( Widget w,

```


[illegible]

```

{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionPolygon ( w, callData );
}

void BbLROIUI::doOptionRectangleCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionRectangle ( w, callData );
}

void BbLROIUI::doOptionRedCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionRed ( w, callData );
}

void BbLROIUI::doOptionWhiteCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionWhite ( w, callData );
}

void BbLROIUI::doOptionYellowCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbLROIUI* obj = ( BbLROIUI * ) clientData;
    obj->doOptionYellow ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbLROIUI::doButtonAccept ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doButtonHide ( Widget, XtPointer )
{
    // This virtual function is called from doButtonHideCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doButtonShow ( Widget, XtPointer )
{
    // This virtual function is called from doButtonShowCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionBlack ( Widget, XtPointer )

```

```

{
    // This virtual function is called from doOptionBlackCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionBlue ( Widget, XtPointer )
{
    // This virtual function is called from doOptionBlueCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionDraw ( Widget, XtPointer )
{
    // This virtual function is called from doOptionDrawCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionEllipse ( Widget, XtPointer )
{
    // This virtual function is called from doOptionEllipseCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionEraseLeft ( Widget, XtPointer )
{
    // This virtual function is called from doOptionEraseLeftCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionEraseRight ( Widget, XtPointer )
{
    // This virtual function is called from doOptionEraseRightCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionFreeHand ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFreeHandCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionGreen ( Widget, XtPointer )
{
    // This virtual function is called from doOptionGreenCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionModify ( Widget, XtPointer )
{
    // This virtual function is called from doOptionModifyCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionPolygon ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPolygonCallback.
    // This function is normally overridden by a derived class.
}

```

```

}

void BbLROIUI::doOptionRectangle ( Widget, XtPointer )
{
    // This virtual function is called from doOptionRectangleCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionRed ( Widget, XtPointer )
{
    // This virtual function is called from doOptionRedCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionWhite ( Widget, XtPointer )
{
    // This virtual function is called from doOptionWhiteCallback.
    // This function is normally overridden by a derived class.
}

void BbLROIUI::doOptionYellow ( Widget, XtPointer )
{
    // This virtual function is called from doOptionYellowCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbLWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "BbLWaveform.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

```

```
extern void VkUnimplemented ( Widget, const char * );
```

```

////////////////////////////////////
// The following non-container elements are created by BbLWaveformUI and are
// available as protected data members inherited by this class
//
// XmLabel          _labelUnit1
// XmLabel          _labelCurrentNum1
// XmLabel          _labelMinNum2
// XmLabel          _labelMaxNum2
// XmList           _scrolledListVessel
// VkOptionMenu *   _optionMenuFlow
// VkMenuItem *     _optionVFR
// VkMenuItem *     _optionPSV
// VkMenuItem *     _optionBSV
// VkMenuItem *     _optionASV
// VkMenuItem *     _optionArea
//
////////////////////////////////////

```

```
///----- Start editable code block: headers and declarations
```

```

#include "Utility.h"
#include "Utility_Widget.h"

```

```
///----- End editable code block: headers and declarations
```

```
///----- BbLWaveform Constructor
```

```

BbLWaveform::BbLWaveform(const char *name, Widget parent)
    BbLWaveformUI(name, parent)
{
    // This constructor calls BbLWaveformUI(parent, name)
    // which calls BbLWaveformUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: BbLWaveform constructor

    _current_vessel = 0;

    //----- End editable code block: BbLWaveform constructor

} // End Constructor

BbLWaveform::BbLWaveform(const char *name) :
    BbLWaveformUI(name)
{
    // This constructor calls BbLWaveformUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //----- Start editable code block: BbLWaveform constructor 2

    _current_vessel = 0;

    //----- End editable code block: BbLWaveform constructor 2

} // End Constructor

BbLWaveform::~BbLWaveform()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //----- Start editable code block: BbLWaveform destructor

    //----- End editable code block: BbLWaveform destructor

} // End Destructor

const char * BbLWaveform::className() // classname
{
    return ("BbLWaveform");
} // End className()

void BbLWaveform::doOptionASV ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbLWaveform doOptionASV

```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbLWaveform::doOptionASV is implemented:
//::VkUnimplemented ( w, "BbLWaveform::doOptionASV" );

_objMag -> msgsLeft.flow_select = FLOW_MV;
_objMag -> update_Lwave(_current_vessel);

//---- End editable code block: BbLWaveform doOptionASV
} // End BbLWaveform::doOptionASV()

void BbLWaveform::doOptionArea ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLWaveform doOptionArea

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLWaveform::doOptionArea is implemented:
    //::VkUnimplemented ( w, "BbLWaveform::doOptionArea" );

    _objMag -> msgsLeft.flow_select = FLOW_AREA;
    _objMag -> update_Lwave(_current_vessel);

    //---- End editable code block: BbLWaveform doOptionArea
} // End BbLWaveform::doOptionArea()

void BbLWaveform::doOptionBSV ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLWaveform doOptionBSV

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLWaveform::doOptionBSV is implemented:
    //::VkUnimplemented ( w, "BbLWaveform::doOptionBSV" );

    _objMag -> msgsLeft.flow_select = FLOW_BSV;
    _objMag -> update_Lwave(_current_vessel);

    //---- End editable code block: BbLWaveform doOptionBSV
} // End BbLWaveform::doOptionBSV()

void BbLWaveform::doOptionPSV ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLWaveform doOptionPSV

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLWaveform::doOptionPSV is implemented:
    //::VkUnimplemented ( w, "BbLWaveform::doOptionPSV" );

    _objMag -> msgsLeft.flow_select = FLOW_PSV;
    _objMag -> update_Lwave(_current_vessel);
}

```

```

//---- End editable code block: BbLWaveform doOptionPSV
} // End BbLWaveform::doOptionPSV()

void BbLWaveform::doOptionVFR ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLWaveform doOptionVFR

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbLWaveform::doOptionVFR is implemented:
    //::VkUnimplemented ( w, "BbLWaveform::doOptionVFR" );

    _objMag -> msgsLeft.flow_select = FLOW_VFR;
    printf(" Current Vessel:  %d\n", _current_vessel);

    _objMag -> update_Lwave(_current_vessel);

    //---- End editable code block: BbLWaveform doOptionVFR
} // End BbLWaveform::doOptionVFR()

void BbLWaveform::vesselLWaveform ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbLWaveform vesselLWaveform

    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbLWaveform::vesselLWaveform is implement
    //::VkUnimplemented ( w, "BbLWaveform::vesselLWaveform" );

    for(int i=0; i<_objMag ->_num_vessels; i++)
        if( XmListPosSelected(w, i) ) break;

    if(i==0) i = _objMag ->_num_vessels - 1;
    else --i;

    _current_vessel = i;
    _objMag -> _vessel = _current_vessel;

    //---- End editable code block: BbLWaveform vesselLWaveform
} // End BbLWaveform::vesselLWaveform()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *BbLWaveform::CreateBbLWaveform( const char *name, Widget parent )
{

```



```

    VkComponent *obj = new BbLWaveform ( name, parent );
    return ( obj );
} // End CreateBbLWaveform

```

```

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

```

```

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbLWaveform::RegisterBbLWaveformInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int                (Use XmRInt)
    //     float             (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg")
    //     A filename        (Use VkRFilename or "Filename")
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback        (Use XmRCallback)

```

```

static InterfaceMap map[] = {
//---- Start editable code block: BbLWaveformUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbLWaveformUI resource table
    { NULL }, // MUST be NULL terminated
};

```

```

    return map;
} // End RegisterBbLWaveformInterface()

```

```

//---- End of generated code

```

```

//---- Start editable code block: End of generated code

```

```
void BbLWaveform::set_unit(char *str)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelUnit1, str);
    delete uw;
}

void BbLWaveform::set_info(float minI, float maxI, float avg)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelMinNum2, minI);
    uw -> set_label(_labelMaxNum2, maxI);
    uw -> set_label(_labelCurrentNum1, avg);
    delete uw;
}

void BbLWaveform::add_vessel(char *str)
{
    XmString item = XmStringCreateSimple(str);
    XmListAddItem(_scrolledListVessel, item, _objMag -> _num_vessels);
}

//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbLWaveformUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbLWaveformUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbLWaveformUI::_defaultBbLWaveformUIResources[] = {
    "*labelCurrentNum1.labelString: 0",
    "*labelMaxNum2.labelString: 1",
    "*labelMinNum2.labelString: 0",
    "*labelUnit1.labelString: Unit",
    "*optionASV.labelString: ASV",
    "*optionArea.labelString: Area",
    "*optionBSV.labelString: BSV",
    "*optionMenuFlow.labelString: ",
    "*optionPSV.labelString: PSV",
    "*optionVFR.labelString: VFR",
    "*tabLabel: Waveform",

    //---- Start editable code block: BbLWaveformUI Default Resources

    //---- End editable code block: BbLWaveformUI Default Resources

    (char*)NULL
};

BbLWaveformUI::BbLWaveformUI ( const char *name ) : VkComponent ( name )

```

```

{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbLWaveform constructor 2

    //---- End editable code block: BbLWaveform constructor 2

} // End Constructor

BbLWaveformUI::BbLWaveformUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //---- Start editable code block: BbLWaveform pre-create

    //---- End editable code block: BbLWaveform pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: BbLWaveform constructor

    //---- End editable code block: BbLWaveform constructor

} // End Constructor

BbLWaveformUI::~BbLWaveformUI()
{
    // Base class destroys widgets

    //---- Start editable code block: BbLWaveformUI destructor

    //---- End editable code block: BbLWaveformUI destructor
} // End destructor

void BbLWaveformUI::create ( Widget parent )
{
    Arg      args[7];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBbLWaveformUIResources );

    // Create an unmanaged widget as the top of the widget hierarchy
    _baseWidget = _bbLWaveform = XtVaCreateWidget ( _name,

```

```

xmButtonInBoardWidgetClass,
parent = 430
XmNresizePolicy, XmRESIZE_GROW,
(XtPointer) NULL );

```

```
// install a callback to guard against unexpected widget destruction
```

```
// Create widgets used in this component
// All variables are data members of this class
```



```

{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->doOptionASV ( w, callData );
}

void BbLWaveformUI::doOptionAreaCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->doOptionArea ( w, callData );
}

void BbLWaveformUI::doOptionBSVCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->doOptionBSV ( w, callData );
}

void BbLWaveformUI::doOptionPSVCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->doOptionPSV ( w, callData );
}

void BbLWaveformUI::doOptionVFRCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->doOptionVFR ( w, callData );
}

void BbLWaveformUI::vesselLWaveformCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbLWaveformUI* obj = ( BbLWaveformUI * ) clientData;
    obj->vesselLWaveform ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbLWaveformUI::doOptionASV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionASVCallback.
    // This function is normally overridden by a derived class.
}

void BbLWaveformUI::doOptionArea ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAreaCallback.
    // This function is normally overridden by a derived class.
}

```

```
void BbLWaveformUI::doOptionBSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionBSVCallback.
    // This function is normally overridden by a derived class.
}

void BbLWaveformUI::doOptionPSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPSVCallback.
    // This function is normally overridden by a derived class.
}

void BbLWaveformUI::doOptionVFR ( Widget, XtPointer )
{
    // This virtual function is called from doOptionVFRCallback.
    // This function is normally overridden by a derived class.
}

void BbLWaveformUI::vesselLWaveform ( Widget, XtPointer )
{
    // This virtual function is called from vesselLWaveformCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```


User: meide
Host: phoenix
Class: phoenix
Job: BbLROI.C

434

```

////////////////////////////////////
//
// Source file for BbRHistogram
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRHistogramUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbRHistogram.h"
#include <Vk/VkEZ.h>
#include <Sgm/Dial.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbRHistogramUI and are
// available as protected data members inherited by this class
//
// XmLabel                _labelMap8
// XmLabel                _labelMap6
// XmLabel                _labelMap4
// XmLabel                _labelMap2
// XmLabel                _labelMap9
// XmLabel                _labelMap7
// XmLabel                _labelMap5
// XmLabel                _labelMap3
// XmLabel                _labelMap1
// VkOptionMenu *        _optionMenuRhyst
// VkMenuItem *          _optionUpdate
// VkMenuItem *          _optionCoarse
// VkMenuItem *          _optionFine
// VkMenuItem *          _optionROI1
// VkMenuItem *          _optionMapping
// XmLabel                _labelLHistoMin2
// XmLabel                _labelLHistoLow2
// XmLabel                _labelLHistoHigh2
// XmLabel                _labelLHistoMax2
// VkOptionMenu *        _optionMenuLHistogram21
// VkMenuItem *          _optionBlackFlow
// VkMenuItem *          _optionWhiteFlow
// VkMenuItem *          _optionAneurysmFlow
// VkMenuItem *          _optionHead1
// VkMenuItem *          _optionBone1
// VkMenuItem *          _optionLung1

```

```

// VkMenuItem *          _optionSpinel
// VkMenuItem *          _optionAbdomen1
// VkMenuItem *          _optionMediaStinum1
// SgDial                 _dialWidth2
// SgDial                 _dialCenter2
//
////////////////////////////////////

//----- Start editable code block: headers and declarations

#include <math.h>
#include "Utility.h"
#include "Utility_Math.h"

//----- End editable code block: headers and declarations

//----- BbRHistogram Constructor

BbRHistogram::BbRHistogram(const char *name, Widget parent) :
    BbRHistogramUI(name, parent)
{
    // This constructor calls BbRHistogramUI(parent, name)
    // which calls BbRHistogramUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //----- Start editable code block: BbRHistogram constructor

    init();
    //----- End editable code block: BbRHistogram constructor

}    // End Constructor

BbRHistogram::BbRHistogram(const char *name) :
    BbRHistogramUI(name)
{
    // This constructor calls BbRHistogramUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //----- Start editable code block: BbRHistogram constructor 2

    init();
    //----- End editable code block: BbRHistogram constructor 2

}    // End Constructor

BbRHistogram::~BbRHistogram()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //----- Start editable code block: BbRHistogram destructor

```

```

//---- End editable code block: BbRHistogram destructor

} // End Destructor

const char * BbRHistogram::className() // classname
{
    return ("BbRHistogram");
} // End className()

void BbRHistogram::AneurysmFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram AneurysmFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::AneurysmFlow is implemented
    //::VkUnimplemented ( w, "BbRHistogram::AneurysmFlow" );

    update_lowhigh(-150.0, 150.0);

    //---- End editable code block: BbRHistogram AneurysmFlow
} // End BbRHistogram::AneurysmFlow()

void BbRHistogram::BlackFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram BlackFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::BlackFlow is implemented:
    //::VkUnimplemented ( w, "BbRHistogram::BlackFlow" );

    //_objMag -> msgsRight.flowDir = -1;
    //_objMag -> msgsRight.flowDir2 = -1;
    update_lowhigh(-150.0, 0.0);

    //---- End editable code block: BbRHistogram BlackFlow
} // End BbRHistogram::BlackFlow()

void BbRHistogram::WhiteFlow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram WhiteFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::WhiteFlow is implemented:
    //::VkUnimplemented ( w, "BbRHistogram::WhiteFlow" );

    //_objMag -> msgsRight.flowDir = 1;

```

```
//_objMag -> msgsRightFlowDir2 = 1;  
update_lowhigh(0.0, 1.0);
```

438

```
//---- End editable code block: BbRHistogram WhiteFlow  
}  
// End BbRHistogram::WhiteFlow()
```

```
void BbRHistogram::abdomen2 ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbRHistogram abdomen2  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when BbRHistogram::abdomen2 is implemented:  
    ::VkUnimplemented ( w, "BbRHistogram::abdomen2" );
```

```
//---- End editable code block: BbRHistogram abdomen2  
}  
// End BbRHistogram::abdomen2()
```

```
void BbRHistogram::bone2 ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbRHistogram bone2  
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;  
    //--- Comment out the following line when BbRHistogram::bone2 is implemented:  
    ::VkUnimplemented ( w, "BbRHistogram::bone2" );
```

```
//---- End editable code block: BbRHistogram bone2  
}  
// End BbRHistogram::bone2()
```

```
void BbRHistogram::centerDrag2 ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbRHistogram centerDrag2  
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;  
    //--- Comment out the following line when BbRHistogram::centerDrag2 is implemented:  
    //::VkUnimplemented ( w, "BbRHistogram::centerDrag2" );
```

```
//---- End editable code block: BbRHistogram centerDrag2  
}  
// End BbRHistogram::centerDrag2()
```

```
void BbRHistogram::doOptionCoarse ( Widget w, XtPointer callData )  
{  
    //---- Start editable code block: BbRHistogram doOptionCoarse
```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData#39

//--- Comment out the following line when BbRHistogram::doOptionCoarse is implement
//::VkUnimplemented ( w, "BbRHistogram::doOptionCoarse" );

_objMag -> msgsRight.histo_status = HISTOGRAM_COARSE;
_objMag -> update_Rhisto();

//---- End editable code block: BbRHistogram doOptionCoarse
} // End BbRHistogram::doOptionCoarse()

void BbRHistogram::doOptionFine ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram doOptionFine
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::doOptionFine is implemented
    //::VkUnimplemented ( w, "BbRHistogram::doOptionFine" );

    _objMag -> msgsRight.histo_status = HISTOGRAM_FINE;
    _objMag -> update_Rhisto2();

    //---- End editable code block: BbRHistogram doOptionFine
} // End BbRHistogram::doOptionFine()

void BbRHistogram::doOptionMapping ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram doOptionMapping
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::doOptionMapping is implemer
    //::VkUnimplemented ( w, "BbRHistogram::doOptionMapping" );

    _objMag -> msgsRight.histo_status = HISTOGRAM_MAPPING;
    _objMag -> update_RhistoMapping();

    //---- End editable code block: BbRHistogram doOptionMapping
} // End BbRHistogram::doOptionMapping()

void BbRHistogram::doOptionROI ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram doOptionROI
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::doOptionROI is implemented:
    //::VkUnimplemented ( w, "BbRHistogram::doOptionROI" );

    _objMag -> msgsRight.histo_status = HISTOGRAM_ROI;

```

```
//---- End editable code block: BbRHistogram doOptionROI
} // End BbRHistogram::doOptionROI()

void BbRHistogram::doOptionUpdate ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram doOptionUpdate
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbRHistogram::doOptionUpdate is implement
    ::VkUnimplemented ( w, "BbRHistogram::doOptionUpdate" );

    //---- End editable code block: BbRHistogram doOptionUpdate
} // End BbRHistogram::doOptionUpdate()

void BbRHistogram::head2 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram head2
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbRHistogram::head2 is implemented:
    ::VkUnimplemented ( w, "BbRHistogram::head2" );

    //---- End editable code block: BbRHistogram head2
} // End BbRHistogram::head2()

void BbRHistogram::highChg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram highChg
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;
    //--- Comment out the following line when BbRHistogram::highChg is implemented:
    //::VkUnimplemented ( w, "BbRHistogram::highChg" );

    int tmp;
    SgDialGetValue(w, &tmp);
    if(_winWidth == -1)
        _winWidth = tmp;
    else
    {
        int x = tmp - _winWidth;
        if(fabsf(x) < 200)
            update_width(x);
        _winWidth = tmp;
    }
}
```

```

//---- End editable code block: BbRHistogram highChg
}    // End BbRHistogram::highChg()

void BbRHistogram::lowChg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram lowChg

    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::lowChg is implemented:
    //::VkUnimplemented ( w, "BbRHistogram::lowChg" );

    int tmp;
    SgDialGetValue(w, &tmp);
    if(_winCenter == -1)
        _winCenter = tmp;
    else
    {
        int x = tmp - _winCenter;
        if(fabsf(x) < 200)
            update_center(x);
        _winCenter = tmp;
    }

    //---- End editable code block: BbRHistogram lowChg
}    // End BbRHistogram::lowChg()

void BbRHistogram::lung2 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram lung2

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::lung2 is implemented:
    ::VkUnimplemented ( w, "BbRHistogram::lung2" );

    //---- End editable code block: BbRHistogram lung2
}    // End BbRHistogram::lung2()

void BbRHistogram::mediastinum2 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram mediastinum2

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRHistogram::mediastinum2 is implemented
    ::VkUnimplemented ( w, "BbRHistogram::mediastinum2" );
}

```



```

//---- End editable code block: BbRHistogram mediastinum2
} // End BbRHistogram::mediastinum2()

void BbRHistogram::spine2 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram spine2
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbRHistogram::spine2 is implemented:
    ::VkUnimplemented ( w, "BbRHistogram::spine2" );

    //---- End editable code block: BbRHistogram spine2
} // End BbRHistogram::spine2()

void BbRHistogram::widthDrag2 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRHistogram widthDrag2
    SgDialCallbackStruct *cbs = (SgDialCallbackStruct*) callData;
    //--- Comment out the following line when BbRHistogram::widthDrag2 is implemented:
    ::VkUnimplemented ( w, "BbRHistogram::widthDrag2" );

    //---- End editable code block: BbRHistogram widthDrag2
} // End BbRHistogram::widthDrag2()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *BbRHistogram::CreateBbRHistogram( const char *name, Widget parent )
{
    VkComponent *obj = new BbRHistogram ( name, parent );
    return ( obj );
} // End CreateBbRHistogram

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbRHistogram::RegisterBbRHistogramInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int               (Use XmRInt)
    //     float             (Use XmRFloat)
    //     No argument       (Use VkRNoArg or "NoArg")
    //     A filename        (Use VkRFilename or "Filename")
    //     An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbRHistogramUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbRHistogramUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbRHistogramInterface()

```

```

//---- End of generated code

```

```

//---- Start editable code block: End of generated code

```

```

void BbRHistogram::init()
{
    _winCenter = -1;
    _winWidth = -1;
}

void BbRHistogram::update(float c, float w)
{
    _objMag -> msgsRight.img_winCenter = c - w/2;
    _objMag -> msgsRight.img_winWidth = c + w/2;
    _objMag -> update_RimgView(_objMag -> msgsRight.img_winCenter,
        _objMag -> msgsRight.img_winWidth);
}

```

```

}

void BbRHistogram::update_lowhigh(float low, float high)
{
    _objMag -> msgsRight.img_winCenter = low;
    _objMag -> msgsRight.img_winWidth = high;
    _objMag -> update_RimgView(_objMag -> msgsRight.img_winCenter,
        _objMag -> msgsRight.img_winWidth);
}

void BbRHistogram::update_width(int x)
{
    _objMag -> msgsRight.img_winCenter -= float(x)/2.0;
    _objMag -> msgsRight.img_winWidth += float(x)/2.0;
    _objMag -> update_RimgView(_objMag -> msgsRight.img_winCenter,
        _objMag -> msgsRight.img_winWidth);
}

void BbRHistogram::update_center(int x)
{
    _objMag -> msgsRight.img_winCenter += float(x);
    _objMag -> msgsRight.img_winWidth += float(x);
    _objMag -> update_RimgView(_objMag -> msgsRight.img_winCenter,
        _objMag -> msgsRight.img_winWidth);
}

void BbRHistogram::set_mapLabels()
{
    float low, high;
    Utility_Widget *uw = new Utility_Widget();
    Utility_Math *um = new Utility_Math();

    if(_objMag -> msgsRight.flowDir >= 0)
    {
        low = _objMag -> msgsRight.img_winCenter;
        high = _objMag -> msgsRight.img_winWidth;
        float dx = (high - low)/6.0;
        uw -> set_label(_labelMap1, um->int_t(low-200));
        uw -> set_label(_labelMap2, um->int_t(low));
        uw -> set_label(_labelMap3, um->int_t(low+dx));
        uw -> set_label(_labelMap4, um->int_t(low+2.0*dx));
        uw -> set_label(_labelMap5, um->int_t(low+3.0*dx));
        uw -> set_label(_labelMap6, um->int_t(low+4.0*dx));
        uw -> set_label(_labelMap7, um->int_t(low+5.0*dx));
        uw -> set_label(_labelMap8, um->int_t(low+6.0*dx));
        uw -> set_label(_labelMap9, um->int_t(high+168.0));
    }
    else
    {
        low = _objMag -> msgsRight.img_winCenter;
        high = _objMag -> msgsRight.img_winWidth;
        float dx = (high - low)/6.0;
        uw -> set_label(_labelMap1, um->int_t(low-168.0));
        uw -> set_label(_labelMap2, um->int_t(low));
        uw -> set_label(_labelMap3, um->int_t(low+dx));
        uw -> set_label(_labelMap4, um->int_t(low+2.0*dx));
        uw -> set_label(_labelMap5, um->int_t(low+3.0*dx));
        uw -> set_label(_labelMap6, um->int_t(low+4.0*dx));
        uw -> set_label(_labelMap7, um->int_t(low+5.0*dx));
        uw -> set_label(_labelMap8, um->int_t(low+6.0*dx));
        uw -> set_label(_labelMap9, um->int_t(high+200.0));
    }
    delete um;
    delete uw;
}

```

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbRHistogramUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbRHistogramUI.h" // Generated header file for this class

#include <Sgm/Dial.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbRHistogramUI::_defaultBbRHistogramUIResources[] = {
    "*labelLHistoHigh2.labelString: 1",
    "*labelLHistoLow2.labelString: 0",
    "*labelLHistoMax2.labelString: 1",
    "*labelLHistoMin2.labelString: 0",
    "*labelMap1.labelString: ",
    "*labelMap2.labelString: ",
    "*labelMap3.labelString: ",
    "*labelMap4.labelString: ",
    "*labelMap5.labelString: ",
    "*labelMap6.labelString: ",
    "*labelMap7.labelString: ",
    "*labelMap8.labelString: ",
    "*labelMap9.labelString: ",
    "*optionAbdomen1.labelString: Abdomen",
    "*optionAneurysmFlow.labelString: Aneurysm Flow",
    "*optionBlackFlow.labelString: Black Flow",
    "*optionBone1.labelString: Bone",
    "*optionCoarse.labelString: Coarse",
    "*optionFine.labelString: Fine",
    "*optionHead1.labelString: Head",
    "*optionLung1.labelString: Lung",
    "*optionMapping.labelString: Mapping",

```

```

    "*optionMediaStim1.labelString: MediaStinum",
    "*optionMenuLHist1.labelString: ",
    "*optionMenuRhHist1.labelString: ",
    "*optionROI1.labelString: ROI",
    "*optionSpine1.labelString: Spine",
    "*optionUpdate.labelString: Update",
    "*optionWhiteFlow.labelString: White Flow",
    "*tabLabel: View",
    "+labelMap1.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap2.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap3.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap4.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap5.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap6.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap7.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap8.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelMap9.fontList: SGI_DYNAMIC SmallPlainLabelFont",

    //---- Start editable code block: BbRHistogramUI Default Resources

    //---- End editable code block: BbRHistogramUI Default Resources

    (char*)NULL
};

BbRHistogramUI::BbRHistogramUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbRHistogram constructor 2

    //---- End editable code block: BbRHistogram constructor 2

}    // End Constructor

BbRHistogramUI::BbRHistogramUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //---- Start editable code block: BbRHistogram pre-create

    //---- End editable code block: BbRHistogram pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: BbRHistogram constructor

    //---- End editable code block: BbRHistogram constructor

}    // End Constructor

```

[illegible]

```
XmNx, 150,
XmNy, 120,
XmNwidth, 4,
XmNheight, 4,
(XtPointer) NULL );
```

```
_labelMap2 = XtVaCreateManagedWidget ( "labelMap2",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 50,
                                         XmNy, 120,
                                         XmNwidth, 4,
                                         XmNheight, 4,
                                         (XtPointer) NULL );
```

```
_labelMap9 = XtVaCreateManagedWidget ( "labelMap9",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 384,
                                         XmNy, 49,
                                         XmNwidth, 4,
                                         XmNheight, 4,
                                         (XtPointer) NULL );
```

```
_labelMap7 = XtVaCreateManagedWidget ( "labelMap7",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 291,
                                         XmNy, 50,
                                         XmNwidth, 4,
                                         XmNheight, 4,
                                         (XtPointer) NULL );
```

```
_labelMap5 = XtVaCreateManagedWidget ( "labelMap5",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 200,
                                         XmNy, 50,
                                         XmNwidth, 4,
                                         XmNheight, 4,
                                         (XtPointer) NULL );
```

```
_labelMap3 = XtVaCreateManagedWidget ( "labelMap3",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 90,
                                         XmNy, 50,
                                         XmNwidth, 4,
                                         XmNheight, 4,
                                         (XtPointer) NULL );
```

[illegible]

[illegible]

```

_baseWidget
XmNx, 432,
XmNy, 92,
XmNwidth, 70,
XmNheight, 60,
(XtPointer) NULL );

```

```

XtAddCallback ( _dialCenter2,
                XmNdragCallback,
                &BbRHistogramUI::centerDrag2Callback,
                (XtPointer) this );

```

```

XtAddCallback ( _dialCenter2,
                XmNvalueChangedCallback,
                &BbRHistogramUI::lowChgCallback,
                (XtPointer) this );

```

```

XtVaSetValues ( _optionMenuRhlist->baseWidget(),
                XmNx, 459,
                XmNy, 10,
                XmNwidth, 122,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

XtVaSetValues ( _optionMenuLHistogram21->baseWidget(),
                XmNx, 417,
                XmNy, 50,
                XmNwidth, 165,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

//---- Start editable code block: BbRHistogramUI create

```

```

//---- End editable code block: BbRHistogramUI create

```

```

}

```

```

const char * BbRHistogramUI::className()
{
    return ("BbRHistogramUI");
} // End className()

```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void BbRHistogramUI::AneurysmFlowCallback ( Widget w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->AneurysmFlow ( w, callData );
}

```

```

void BbRHistogramUI::BlackFlowCallback ( Widget w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->BlackFlow ( w, callData );
}

```

```

void BbRHistogramUI::WhiteFlowCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->WhiteFlow ( w, callData );
}

void BbRHistogramUI::abdomen2Callback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->abdomen2 ( w, callData );
}

void BbRHistogramUI::bone2Callback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->bone2 ( w, callData );
}

void BbRHistogramUI::centerDrag2Callback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->centerDrag2 ( w, callData );
}

void BbRHistogramUI::doOptionCoarseCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionCoarse ( w, callData );
}

void BbRHistogramUI::doOptionFineCallback ( Widget      w,
                                            XtPointer clientData,
                                            XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionFine ( w, callData );
}

void BbRHistogramUI::doOptionMappingCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionMapping ( w, callData );
}

void BbRHistogramUI::doOptionROIcallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionROI ( w, callData );
}

void BbRHistogramUI::doOptionUpdateCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionUpdate ( w, callData );
}

```

```

{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->doOptionUpdate ( w, callData );
}

void BbRHistogramUI::head2Callback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->head2 ( w, callData );
}

void BbRHistogramUI::highChgCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->highChg ( w, callData );
}

void BbRHistogramUI::lowChgCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->lowChg ( w, callData );
}

void BbRHistogramUI::lung2Callback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->lung2 ( w, callData );
}

void BbRHistogramUI::mediastinum2Callback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->mediastinum2 ( w, callData );
}

void BbRHistogramUI::spine2Callback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->spine2 ( w, callData );
}

void BbRHistogramUI::widthDrag2Callback ( Widget      w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbRHistogramUI* obj = ( BbRHistogramUI * ) clientData;
    obj->widthDrag2 ( w, callData );
}

```

```

////////////////////////////////////
// The following functions are called from the menu items

```

```
// in this window.  
////////////////////////////////////
```

453

```
void BbRHistogramUI::AneurysmFlow ( Widget, XtPointer )  
{  
    // This virtual function is called from AneurysmFlowCallback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::BlackFlow ( Widget, XtPointer )  
{  
    // This virtual function is called from BlackFlowCallback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::WhiteFlow ( Widget, XtPointer )  
{  
    // This virtual function is called from WhiteFlowCallback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::abdomen2 ( Widget, XtPointer )  
{  
    // This virtual function is called from abdomen2Callback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::bone2 ( Widget, XtPointer )  
{  
    // This virtual function is called from bone2Callback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::centerDrag2 ( Widget, XtPointer )  
{  
    // This virtual function is called from centerDrag2Callback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::doOptionCoarse ( Widget, XtPointer )  
{  
    // This virtual function is called from doOptionCoarseCallback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::doOptionFine ( Widget, XtPointer )  
{  
    // This virtual function is called from doOptionFineCallback.  
    // This function is normally overridden by a derived class.  
}  
  
void BbRHistogramUI::doOptionMapping ( Widget, XtPointer )  
{  
    // This virtual function is called from doOptionMappingCallback.  
    // This function is normally overridden by a derived class.  
}
```

```

void BbRHistogramUI::doOptionROI ( Widget, XtPointer )
{
    // This virtual function is called from doOptionROICallback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::doOptionUpdate ( Widget, XtPointer )
{
    // This virtual function is called from doOptionUpdateCallback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::head2 ( Widget, XtPointer )
{
    // This virtual function is called from head2Callback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::highChg ( Widget, XtPointer )
{
    // This virtual function is called from highChgCallback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::lowChg ( Widget, XtPointer )
{
    // This virtual function is called from lowChgCallback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::lung2 ( Widget, XtPointer )
{
    // This virtual function is called from lung2Callback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::mediastinum2 ( Widget, XtPointer )
{
    // This virtual function is called from mediastinum2Callback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::spine2 ( Widget, XtPointer )
{
    // This virtual function is called from spine2Callback.
    // This function is normally overridden by a derived class.
}

void BbRHistogramUI::widthDrag2 ( Widget, XtPointer )
{
    // This virtual function is called from widthDrag2Callback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

```



```
XmNwidth, 4  
XmNheight, 4,  
(XtPointer) NULL );
```

456

```
_labelMap7 = XtVaCreateManagedWidget ( "labelMap7",  
                                         xmLabelWidgetClass,  
                                         _baseWidget,  
                                         XmNlabelType, XmSTRING,  
                                         XmNx, 291,  
                                         XmNy, 50,  
                                         XmNwidth, 4,  
                                         XmNheight, 4,  
                                         (XtPointer) NULL );
```

```
_labelMap5 = XtVaCreateManagedWidget ( "labelMap5",  
                                         xmLabelWidgetClass,  
                                         _baseWidget,  
                                         XmNlabelType, XmSTRING,  
                                         XmNx, 200,  
                                         XmNy, 50,  
                                         XmNwidth, 4,  
                                         XmNheight, 4,  
                                         (XtPointer) NULL );
```

```
_labelMap3 = XtVaCreateManagedWidget ( "labelMap3",  
                                         xmLabelWidgetClass,  
                                         _baseWidget,  
                                         XmNlabelType, XmSTRING,  
                                         XmNx, 90,  
                                         XmNy, 50,  
                                         XmNwidth, 4,  
                                         XmNheight, 4,  
                                         (XtPointer) NULL );
```

```
_labelMap1 = XtVaCreateManagedWidget ( "labelMap1",  
                                         xmLabelWidgetClass,  
                                         _baseWidget,  
                                         XmNlabelType, XmSTRING,  
                                         XmNx, 10,  
                                         XmNy, 50,  
                                         XmNwidth, 4,  
                                         XmNheight, 4,  
                                         (XtPointer) NULL );
```

```
}
```

```
//----- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbRROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbRROI.h"
#include <Vk/VkEZ.h>
#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/ScrolledW.h>
#include <Xm/Separator.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbRROIUI and are
// available as protected data members inherited by this class
//
// VkOptionMenu *           _optionMenu5
// VkMenuItem *             _optionROIFlow
// VkMenuItem *             _optionBackFlow
// XmSeparator              _separator3
// XmArrowButton            _arrow1
// VkOptionMenu *           _optionMenu8
// VkMenuItem *             _optionOpenROI
// VkMenuItem *             _optionSave3D
// VkMenuItem *             _optionShow3D
// VkMenuItem *             _optionShowNeighbor
// VkMenuItem *             _optionShowAllNeighbor
// VkMenuItem *             _optionHideNeighbor
// VkOptionMenu *           _optionMenu7
// VkMenuItem *             _optionShow
// VkMenuItem *             _optionShowAll
// VkMenuItem *             _optionHide
// VkMenuItem *             _optionModifyROI
// VkMenuItem *             _optionModify3D
// XmList                   _scrolledListROIS
// XmPushButton             _buttonSaveROI
// XmPushButton             _buttonAcceptROI

```

```
// XmLabel          _labelNumCurr
// XmLabel          _labelNumROI
// XmArrowButton    _arrowPrevROI
// XmPushButton     _buttonRemoveROI
// XmTextField      _textfieldROIName
// XmLabel          _labelROIName
//
////////////////////////////////////
```

```
//---- Start editable code block: headers and declarations
```

```
#include "ROIS.h"
#include <Vk/VkFormat.h>
```

```
//---- End editable code block: headers and declarations
```

```
//---- BbRROI Constructor
```

```
BbRROI::BbRROI(const char *name, Widget parent) :
    BbRROIUI(name, parent)
{
    // This constructor calls BbRROIUI(parent, name)
    // which calls BbRROIUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbRROI constructor

    //---- End editable code block: BbRROI constructor

}    // End Constructor
```

```
BbRROI::BbRROI(const char *name) :
    BbRROIUI(name)
{
    // This constructor calls BbRROIUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbRROI constructor 2

    //---- End editable code block: BbRROI constructor 2

}    // End Constructor
```

```
BbRROI::~BbRROI()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbRROI destructor
```

```

//----- End editable code block: BbRROI destructor

} // End Destructor

const char * BbRROI::className() // classname
{
    return ("BbRROI");
} // End className()

void BbRROI::NextNeighbor ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRROI NextNeighbor

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::NextNeighbor is implemented:
    //::VkUnimplemented ( w, "BbRROI::NextNeighbor" );

    ++_roi_nn;
    draw_ROINeighbor();

    //----- End editable code block: BbRROI NextNeighbor
} // End BbRROI::NextNeighbor()

void BbRROI::PrevROI ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRROI PrevROI

    XmArrowButtonCallbackStruct *cbs = (XmArrowButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::PrevROI is implemented:
    //::VkUnimplemented ( w, "BbRROI::PrevROI" );

    --_ROI_NO;
    if(_mode == 0)
        draw_ROI();
    else if(_mode == 1)
        modify();

    //----- End editable code block: BbRROI PrevROI
} // End BbRROI::PrevROI()

void BbRROI::ROIName ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRROI ROIName

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::ROIName is implemented:
    //::VkUnimplemented ( w, "BbRROI::ROIName" );

```

```

char name[100];
sprintf(name, "%s", XmTextFieldGetString(_textfieldROIName));
if(_objMag->_imageView2->_ROI != NULL)
{
    Points *p = _objMag->_imageView2->_ROI->_points_in_border.get_Points(_objMag->_imageView2->_ROI->_points_in_border.roi_x, _objMag->_imageView2->_ROI->_points_in_border.roi_y);

    ((ROIS *) (_objMag -> _ROIS)) -> add(_objMag -> msgsRight.img_number -
        _objMag -> msgsLoaded.img_start, name, p);

    //_objMag->_imageView2 -> EraseROI();
}

//---- End editable code block: BbRROI ROIName
} // End BbRROI::ROIName()

void BbRROI::doButtonAcceptROI ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doButtonAcceptROI

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doButtonAcceptROI is implemented:
    //::VkUnimplemented ( w, "BbRROI::doButtonAcceptROI" );

    char name[100];
    sprintf(name, "%s", XmTextFieldGetString(_textfieldROIName));

    if(_objMag->_imageView2->_ROI != NULL)
    {
        Points *p = _objMag->_imageView2->_ROI->_points_in_border.get_Points(_objMag->_imageView2->_ROI->_points_in_border.roi_x, _objMag->_imageView2->_ROI->_points_in_border.roi_y);

        ((ROIS *) (_objMag -> _ROIS)) -> add(_objMag -> msgsRight.img_number -
            _objMag -> msgsLoaded.img_start, name, p);

        //_objMag->_imageView2 -> EraseROI();
    }

    //---- End editable code block: BbRROI doButtonAcceptROI
} // End BbRROI::doButtonAcceptROI()

void BbRROI::doButtonRemove ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doButtonRemove

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doButtonRemove is implemented:
    //::VkUnimplemented ( w, "BbRROI::doButtonRemove" );

    int img_number = _objMag -> msgsRight.img_number -
        _objMag -> msgsLoaded.img_start;
    ((ROIS *) (_objMag -> _ROIS)) -> remove(img_number, _ROI_NO);

    //---- End editable code block: BbRROI doButtonRemove
} // End BbRROI::doButtonRemove()

```

```
void BbRROI::doButtonSave ( Widget w, XtPointer callData )
```

461

```
{
    //---- Start editable code block: BbRROI doButtonSaveROI

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doButtonSaveROI is implemented:
    //::VkUnimplemented ( w, "BbRROI::doButtonSaveROI" );

    ROIS *rois = (ROIS *)(_objMag -> _ROIS);
    rois -> to_File();

    XmTextFieldSetString(_textfieldROIName, "Save Done");

    //---- End editable code block: BbRROI doButtonSaveROI
}    // End BbRROI::doButtonSaveROI()
```

```
void BbRROI::doOptionBackFlow ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbRROI doOptionBackFlow

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionBackFlow is implemented:
    //::VkUnimplemented ( w, "BbRROI::doOptionBackFlow" );

    XmTextFieldSetString(_textfieldROIName, "Back");

    //---- End editable code block: BbRROI doOptionBackFlow
}    // End BbRROI::doOptionBackFlow()
```

```
void BbRROI::doOptionHide ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbRROI doOptionHide

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionHide is implemented:
    //::VkUnimplemented ( w, "BbRROI::doOptionHide" );

    _objMag->_imgView2 -> display();

    //---- End editable code block: BbRROI doOptionHide
}    // End BbRROI::doOptionHide()
```

```
void BbRROI::doOptionHideNeighbor ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbRROI doOptionHideNeighbor

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionHideNeighbor is implement
    //::VkUnimplemented ( w, "BbRROI::doOptionHideNeighbor" );
}
```

```

//---- End editable code block: BbRROI doOptionHideNeighbor
} // End BbRROI::doOptionHideNeighbor()

void BbRROI::doOptionModify ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionModify
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionModify is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionModify" );

 ROI_NO = 0;
 mode = 1;
modify();

//---- End editable code block: BbRROI doOptionModify
} // End BbRROI::doOptionModify()

void BbRROI::doOptionModify3D ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionModify3D
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionModify3D is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionModify3D" );

GE_PCMRA_HEADER_OBJ *pc = _objMag -> _img -> get_header();
_objMag-> _ROIS -> to_ivFile(_objMag -> msgsRight.img_number - _objMag ->
msgsLoaded.img_start, _ROI_NO, pc->slthick, pc->pixsize_X, pc->pixsize_Y);
XmTextFieldSetString(_textfieldROIName, "Modify3D");
_objMag-> update_win3D();

//---- End editable code block: BbRROI doOptionModify3D
} // End BbRROI::doOptionModify3D()

void BbRROI::doOptionOpenROI ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionOpenROI
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionOpenROI is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionOpenROI" );

char str[300];
sprintf(str, "%s/ROIS.DAT", _objMag -> msgsLoaded.img_dir);
_objMag-> _ROIS -> from_File(str);
set_list();
XmTextFieldSetString(_textfieldROIName, "Read dn");

```

```

//---- End editable code block: BbRROI doOptionOpenROI
} // End BbRROI::doOptionOpenROI()

void BbRROI::doOptionROIFlow ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionROIFlow
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionROIFlow is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionROIFlow" );
XmTextFieldSetString(_textfieldROIName, "Flow");

//---- End editable code block: BbRROI doOptionROIFlow
} // End BbRROI::doOptionROIFlow()

void BbRROI::doOptionSave3D ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionSave3D
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionSave3D is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionSave3D" );

GE_PCMRA_HEADER_OBJ *pc = _objMag -> _img -> get_header();
_objMag-> _ROIS -> to_File(pc->slthick, pc->pixsize_X, pc->pixsize_Y);
//_objMag-> _ROIS -> to_ivFileSurface(pc->slthick, pc->pixsize_X, pc->pixsize_Y);
_objMag-> _ROIS -> to_ivFile(pc->slthick, pc->pixsize_X, pc->pixsize_Y);

XmTextFieldSetString(_textfieldROIName, "Save dn");

//---- End editable code block: BbRROI doOptionSave3D
} // End BbRROI::doOptionSave3D()

void BbRROI::doOptionShow ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRROI doOptionShow
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbRROI::doOptionShow is implemented:
//::VkUnimplemented ( w, "BbRROI::doOptionShow" );

 ROI_NO = 0;
_mode = 0;
draw_ROI();

//---- End editable code block: BbRROI doOptionShow
} // End BbRROI::doOptionShow()

```



```

void BbRROI::doOptionShow ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doOptionShow3D

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionShow3D is implemented:
    //::VkUnimplemented ( w, "BbRROI::doOptionShow3D" );

    _objMag-> update_win3D();

    //---- End editable code block: BbRROI doOptionShow3D
}    // End BbRROI::doOptionShow3D()

void BbRROI::doOptionShowAll ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doOptionShowAll

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionShowAll is implemented:
    //::VkUnimplemented ( w, "BbRROI::doOptionShowAll" );

    _mode = 2;
    draw_AllROI(_objMag ->msgsRight.img_number - _objMag ->msgsLoaded.img_start);

    //---- End editable code block: BbRROI doOptionShowAll
}    // End BbRROI::doOptionShowAll()

void BbRROI::doOptionShowAllNeighbor ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doOptionShowAllNeighbor

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionShowAllNeighbor is implem
    //::VkUnimplemented ( w, "BbRROI::doOptionShowAllNeighbor" );

    //draw_AllROINeighbor();

    //---- End editable code block: BbRROI doOptionShowAllNeighbor
}    // End BbRROI::doOptionShowAllNeighbor()

void BbRROI::doOptionShowNeighbor ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI doOptionShowNeighbor

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::doOptionShowNeighbor is implement
    //::VkUnimplemented ( w, "BbRROI::doOptionShowNeighbor" );

```

```

    _roi_nn = 0;
    draw_ROINeighbor();

    //---- End editable code block: BbRROI doOptionShowNeighbor
}    // End BbRROI::doOptionShowNeighbor()

void BbRROI::rois ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRROI rois

    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbRROI::rois is implemented:

    //::VkUnimplemented ( w, "BbRROI::rois" );

    for(int i=0; i<=(_objMag -> msgsLoaded.img_end - _objMag -> msgsLoaded.img_start);
        if( XmListPosSelected(w, i) ) break;

    if(i==0) i = _objMag -> msgsLoaded.img_end - _objMag -> msgsLoaded.img_start;
    else --i;

    _frame = i;

    //---- End editable code block: BbRROI rois
}    // End BbRROI::rois()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *BbRROI::CreateBbRROI( const char *name, Widget parent )
{
    VkComponent *obj = new BbRROI ( name, parent );
    return ( obj );
} // End CreateBbRROI

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbRROI::RegisterBbRROIInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char *      (Use XmRString)
    // Boolean           (Use XmRBoolean)
    // int               (Use XmRInt)
    // float             (Use XmRFloat)
    // No argument       (Use VkrNoArg or "NoArg")
    // A filename        (Use VkrFilename or "Filename")
    // An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    // A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //---- Start editable code block: BbRROIUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //---- End editable code block: BbRROIUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbRROIInterface()

//---- End of generated code

//---- Start editable code block: End of generated code

void BbRROI::draw_ROINeighbor()
{
    ROIS *rois = (ROIS *)(_objMag -> _ROIS);

    show_total(rois->_ROI[_frame]._numROIs);

    if( rois->_ROI[_frame]._numROIs == 0)
    {
        show_current(0);
        XmTextFieldSetString(_textfieldROIName, "");
        return;
    }

    if(_roi_nn < 0) _roi_nn = rois->_ROI[_frame]._numROIs - 1;
    if(_roi_nn >= rois->_ROI[_frame]._numROIs) _roi_nn = 0;

    show_current(_roi_nn + 1);

    ROI_OBJ *roi = &(rois->_ROI[_frame]._ROI_OBJ[_roi_nn]);

    //_objMag->_imgView2 -> display();

```

```

roi -> _points -> inverse_get_Points(_objMag->_imgView2 -> _zoom,
    _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y);
-> draw(_objMag -> _imgView2 -> baseWidget(), _objMag -> _imgView2 -> _roi_color);

XmTextFieldSetString(_textfieldROIName, (char *)VkFormat("%s", roi->_name ));
}

void BbRROI::draw_AllROI(int img_number)
{
    ROIS *rois = (ROIS *)(_objMag -> _ROIS);

    show_total(rois->_ROI[img_number]._numROIs);

    if( rois->_ROI[img_number]._numROIs == 0)
    {
        show_current(0);
        XmTextFieldSetString(_textfieldROIName, "");
        return;
    }

    ROI_OBJ *roi;
    _objMag->_imgView2 -> display();

    for(int i=0; i<rois->_ROI[img_number]._numROIs; i++)
    {
        roi = &(rois->_ROI[img_number]._ROI_OBJ[i]);
        roi -> _points -> inverse_get_Points(_objMag->_imgView2 -> _zoom,
            _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y)
        -> draw(_objMag -> _imgView2 -> baseWidget(), _objMag -> _imgView2 -> _roi_color
    }

    if(rois->_ROI[img_number]._numROIs > 1)
        XmTextFieldSetString(_textfieldROIName, "All");
    else if(rois->_ROI[img_number]._numROIs == 1)
        XmTextFieldSetString(_textfieldROIName, (char *)VkFormat("%s",
            rois->_ROI[img_number]._ROI_OBJ[0]._name ));
    }

void BbRROI::add_AllROI(int number, int number_prev)
{
    ROIS *rois = (ROIS *)(_objMag -> _ROIS);

    if( rois->_ROI[number_prev]._numROIs == 0)
    {
        return;
    }

    ROI_OBJ *roi;

    for(int i=0; i<rois->_ROI[number_prev]._numROIs; i++)
    {
        roi = &(rois->_ROI[number_prev]._ROI_OBJ[i]);
        ((ROIS *)(_objMag -> _ROIS)) -> add(number, roi -> _name, roi -> _points);
    }
}

void BbRROI::draw_ROI()
{
    int img_number = _objMag -> msgsRight.img_number - _objMag -> msgsLoaded.img_start

    ROIS *rois = (ROIS *)(_objMag -> _ROIS);

    printf("BbRROI::draw_ROI (img_number = %d) _numROIs=%d _ROI_NO=%d\n",
        img_number, rois->_ROI[img_number]._numROIs, _ROI_NO);

    show_total(rois->_ROI[img_number]._numROIs);
}

```

```

if( rois->_ROI[img_number]._numROIs == 0)
{
    show_current(0);
    XmTextFieldSetString(_textfieldROIName, "");
    return;
}

if(_ROI_NO < 0) _ROI_NO = rois->_ROI[img_number]._numROIs - 1;
if(_ROI_NO >= rois->_ROI[img_number]._numROIs) _ROI_NO = 0;

show_current(_ROI_NO + 1);

printf("    ROI_No = %d\n", _ROI_NO);
ROI_OBJ *roi = &(rois->_ROI[img_number]._ROI_OBJ[_ROI_NO]);

_objMag->_imgView2 -> display();
roi -> _points -> inverse_get_Points(_objMag->_imgView2 ->_zoom,
    _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y)
    -> draw(_objMag -> _imgView2 -> baseWidget(), _objMag -> _imgView2 -> _roi_color);

/*
_objMag-> _imgView2 -> CreateROI(_objMag-> _imgView2 -> _roi_type);
_objMag-> _imgView2 -> _ROI -> _points_in_border =
    *(roi -> _points -> inverse_get_Points(_objMag->_imgView2 ->_zoom,
        _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y));
_objMag-> _imgView2 -> _ROI -> draw();
*/

XmTextFieldSetString(_textfieldROIName, (char *)VkFormat("%s", roi->_name ));
}

void BbRROI::modify()
{
    int img_number = _objMag -> msgsRight.img_number - _objMag -> msgsLoaded.img_start

    ROIS *rois = (ROIS *)(_objMag -> _ROIS);

    show_total(rois->_ROI[img_number]._numROIs);

    if( rois->_ROI[img_number]._numROIs == 0)
    {
        show_current(0);
        XmTextFieldSetString(_textfieldROIName, "");
        return;
    }

    if(_ROI_NO < 0) _ROI_NO = rois->_ROI[img_number]._numROIs - 1;
    if(_ROI_NO >= rois->_ROI[img_number]._numROIs) _ROI_NO = 0;

    show_current(_ROI_NO + 1);

    ROI_OBJ *roi = &(rois->_ROI[img_number]._ROI_OBJ[_ROI_NO]);

    _objMag-> _imgView2 -> CreateROI2(ROI_FREEHAND);
    _objMag-> _imgView2 -> _ROI -> _points_in_border =
        *(roi -> _points -> inverse_get_Points(_objMag->_imgView2 ->_zoom,
            _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y));

    _objMag-> _imgView2 -> _ROI -> _show_status = TRUE;
    _objMag-> _imgView2 -> _ROI -> _draw_status = TRUE;
    //_objMag-> _imgView2 -> _ROI -> AcceptROI();

    _objMag->_imgView2 -> display();

    //_objMag-> _imgView2 -> _ROI -> draw();
    _objMag-> _imgView2 -> _ROI -> _points_in_border.draw_keyPoints(

```

```
    XmTextFieldSetString(_textfieldROIName, (char *)VkFormat("%s", roi->_name ));
}

int BbRROI::modify(char *name)
{
    int img_number = _objMag -> msgsRight.img_number - _objMag -> msgsLoaded.img_start
    ROIS *rois = (ROIS *)(_objMag -> _ROIS);
    show_total(rois->_ROI[img_number]._numROIs);
    if( rois->_ROI[img_number]._numROIs == 0)
    {
        show_current(0);
        XmTextFieldSetString(_textfieldROIName, "");
        return 0;
    }

    int flag = -1;
    for(int i=0; i<rois->_ROI[img_number]._numROIs; i++)
    {
        if( strcmp(rois->_ROI[img_number]._ROI_OBJ[i]._name, name, 4) == 0)
            flag = i;
    }

    if(flag == -1) return 0;
    else _ROI_NO = flag;

    show_current(flag + 1);

    ROI_OBJ *roi = &(rois->_ROI[img_number]._ROI_OBJ[_ROI_NO]);

    _objMag-> _imgView2 -> CreateROI2(ROI_FREEHAND);
    _objMag-> _imgView2 -> _ROI -> _points_in_border =
        *(roi -> _points -> inverse_get_Points(_objMag->_imgView2 ->_zoom,
        _objMag -> msgsLeft.roi_x, _objMag -> msgsLeft.roi_y));

    _objMag-> _imgView2 -> _ROI -> _show_status = TRUE;
    _objMag-> _imgView2 -> _ROI -> _draw_status = TRUE;
    //_objMag-> _imgView2 -> _ROI -> AcceptROI();

    _objMag->_imgView2 -> display();

    //_objMag-> _imgView2 -> _ROI -> draw();
    //_objMag-> _imgView2 -> _ROI -> _points_in_border.draw_keyPoints(
    //_ _objMag-> _imgView2 -> _ROI -> _widget);

    XmTextFieldSetString(_textfieldROIName, (char *)VkFormat("%s", roi->_name ));

    return 1;
}

void BbRROI::show_current(int i)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelNumCurr, i);
    delete uw;
}

void BbRROI::show_total(int i)
{
    Utility_Widget *uw = new Utility_Widget();
```

```
uw -> set_label(_labelNameROI, i);  
delete uw;  
}
```

470

```
void BbRROI::set_list()  
{  
    XmString item;  
    char str[200];  
  
    ROIS *rois = (ROIS *) (_objMag -> _ROIS);  
    if(rois != NULL)  
    {  
        XmListDeleteAllItems(_scrolledListROIS);  
        for(int i=_objMag -> msgsLoaded.img_start; i<=_objMag -> msgsLoaded.img_end; i++)  
        {  
            sprintf(str, "%d (%d)", i, rois->_ROI[i - _objMag -> msgsLoaded.img_start]._nu  
            item = XmStringCreateSimple(str);  
            XmListAddItem(_scrolledListROIS, item, 0);  
        }  
    }  
}
```

```
//---- End editable code block: End of generated code
```

User: meide
Host: phoenix
Class: phoenix
Job: BbRHistogram.C

471


```

////////////////////////////////////
//
// Source file for BbRROIUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbRROIUI.h" // Generated header file for this class

#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/ScrolledW.h>
#include <Xm/Separator.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbRROIUI::_defaultBbRROIUIResources[] = {
    "*buttonAcceptROI.labelString: Add",
    "*buttonRemoveROI.labelString: Remove",
    "*buttonSaveROI.labelString: Save",
    "*labelNumCurr.labelString: 1",
    "*labelNumROI.labelString: 1",
    "*labelROIName.labelString: ROI Name",
    "*optionBackFlow.labelString: Background",
    "*optionHide.labelString: Hide",
    "*optionHideNeighbor.labelString: Hide",
    "*optionModify3D.labelString: Modify3D",
    "*optionModifyROI.labelString: Modify",
    "*optionOpenROI.labelString: Open ROI",
    "*optionROIFlow.labelString: Vessel ROI",
    "*optionSave3D.labelString: Save 3D",
    "*optionShow.labelString: Show",
    "*optionShow3D.labelString: Show 3D",
    "*optionShowAll.labelString: Show All",

```

```

    "*optionShowAllNeighbors.labelString: Show All",
    "*optionShowNeighbors.labelString: Show",
    "*tabLabel: ROI",

```

473

```

//---- Start editable code block: BbRROIUI Default Resources

```

```

//---- End editable code block: BbRROIUI Default Resources

```

```

(char*)NULL

```

```

};

```

```

BbRROIUI::BbRROIUI ( const char *name ) : VkComponent ( name )
{

```

```

    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

```

```

//---- Start editable code block: BbRROI constructor 2

```

```

//---- End editable code block: BbRROI constructor 2

```

```

} // End Constructor

```

```

BbRROIUI::BbRROIUI ( const char *name, Widget parent ) : VkComponent ( name )
{

```

```

    //---- Start editable code block: BbRROI pre-create

```

```

//---- End editable code block: BbRROI pre-create

```

```

    // Call creation function to build the widget tree.

```

```

    create ( parent );

```

```

//---- Start editable code block: BbRROI constructor

```

```

//---- End editable code block: BbRROI constructor

```

```

} // End Constructor

```

```

BbRROIUI::~BbRROIUI()
{

```

```

    // Base class destroys widgets

```

```

//---- Start editable code block: BbRROIUI destructor

```

```

//---- End editable code block: BbRROIUI destructor

```

```

} // End destructor

```

```

void BbRROIUI::create ( Widget parent )

```

[illegible]

```

_optionSave3D = _opMenu8->addAction ( "optionSave3D",
                                     &BbRROIUI::doOptionSave3DCallback,
                                     (XtPointer) this );

_optionShow3D = _optionMenu8->addAction ( "optionShow3D",
                                     &BbRROIUI::doOptionShow3DCallback,
                                     (XtPointer) this );

_optionShowNeighbor = _optionMenu8->addAction ( "optionShowNeighbor",
                                     &BbRROIUI::doOptionShowNeighborCal
                                     (XtPointer) this );

_optionShowAllNeighbor = _optionMenu8->addAction ( "optionShowAllNeighbor",
                                     &BbRROIUI::doOptionShowAllNeigh
                                     (XtPointer) this );

_optionHideNeighbor = _optionMenu8->addAction ( "optionHideNeighbor",
                                     &BbRROIUI::doOptionHideNeighborCal
                                     (XtPointer) this );

_optionMenu7 = new VkOptionMenu ( _baseWidget, "optionMenu7");
_optionShow = _optionMenu7->addAction ( "optionShow",
                                     &BbRROIUI::doOptionShowCallback,
                                     (XtPointer) this );

_optionShowAll = _optionMenu7->addAction ( "optionShowAll",
                                     &BbRROIUI::doOptionShowAllCallback,
                                     (XtPointer) this );

_optionHide = _optionMenu7->addAction ( "optionHide",
                                     &BbRROIUI::doOptionHideCallback,
                                     (XtPointer) this );

_optionModifyROI = _optionMenu7->addAction ( "optionModifyROI",
                                     &BbRROIUI::doOptionModifyCallback,
                                     (XtPointer) this );

_optionModify3D = _optionMenu7->addAction ( "optionModify3D",
                                     &BbRROIUI::doOptionModify3DCallback,
                                     (XtPointer) this );

_scrolledWindow6 = XtVaCreateManagedWidget ( "scrolledWindow6",
                                     xmScrolledWindowWidgetClass,
                                     _baseWidget,
                                     XmNscrollBarDisplayPolicy, XmSTATIC,
                                     XmNx, 411,
                                     XmNy, 12,
                                     XmNwidth, 150,
                                     XmNheight, 90,
                                     (XtPointer) NULL );

_scrolledListROIS = XtVaCreateManagedWidget ( "scrolledListROIS",
                                     xmListWidgetClass,
                                     _scrolledWindow6,
                                     XmNselectionPolicy, XmBROWSE_SELECT,
                                     XmNlistSizePolicy, XmCONSTANT,
                                     XmNwidth, 144,
                                     XmNheight, 84,
                                     (XtPointer) NULL );

XtAddCallback ( _scrolledListROIS,
               XmNbrowseSelectionCallback,
               &BbRROIUI::roisCallback,
               (XtPointer) this );

```

[illegible]

```
XtAddCallback ( _arrowPrevROI,
                 XmNactivateCallback,
                 &BbRROIUI::PrevROIcallback,
                 (XtPointer) this );
```

```
_buttonRemoveROI = XtVaCreateManagedWidget ( "buttonRemoveROI",
                                                xmPushButtonWidgetClass,
                                                _baseWidget,
                                                XmNlabelType, XmSTRING,
                                                XmNx, 230,
                                                XmNy, 97,
                                                XmNwidth, 90,
                                                XmNheight, 30,
                                                (XtPointer) NULL );
```

```
XtAddCallback ( _buttonRemoveROI,
                 XmNactivateCallback,
                 &BbRROIUI::doButtonRemoveCallback,
                 (XtPointer) this );
```

```
_textfieldROIName = XtVaCreateManagedWidget ( "textfieldROIName",
                                                xmTextFieldWidgetClass,
                                                _baseWidget,
                                                XmNcolumns, 10,
                                                XmNx, 81,
                                                XmNy, 59,
                                                XmNheight, 35,
                                                (XtPointer) NULL );
```

```
XtAddCallback ( _textfieldROIName,
                 XmNactivateCallback,
                 &BbRROIUI::ROINameCallback,
                 (XtPointer) this );
```

```
_labelROIName = XtVaCreateManagedWidget ( "labelROIName",
                                             xmLabelWidgetClass,
                                             _baseWidget,
                                             XmNlabelType, XmSTRING,
                                             XmNx, 90,
                                             XmNy, 28,
                                             XmNwidth, 74,
                                             XmNheight, 20,
                                             (XtPointer) NULL );
```

```
XtVaSetValues ( _optionMenu5->baseWidget(),
                 XmNx, 200,
                 XmNy, 20,
                 XmNwidth, 146,
                 XmNheight, 32,
                 (XtPointer) NULL );
```

```
XtVaSetValues ( _optionMenu8->baseWidget(),
                 XmNx, 402,
                 XmNy, 119,
                 XmNwidth, 129,
                 XmNheight, 32,
                 (XtPointer) NULL );
```

```
XtVaSetValues ( _optionMenu7->baseWidget(),
                 XmNx, 21,
                 XmNy, 110,
                 XmNwidth, 126,
                 XmNheight, 32,
                 (XtPointer) NULL );
```

//---- End editable code block: BbRROIUI create

}

```
const char * BbRROIUI::className()
{
    return ("BbRROIUI");
} // End className()
```

```
////////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////////
```

```
void BbRROIUI::NextNeighborCallback ( Widget    w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->NextNeighbor ( w, callData );
}
```

```
void BbRROIUI::PrevROICallback ( Widget    w,
                                 XtPointer clientData,
                                 XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->PrevROI ( w, callData );
}
```

```
void BbRROIUI::ROINameCallback ( Widget    w,
                                 XtPointer clientData,
                                 XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->ROIName ( w, callData );
}
```

```
void BbRROIUI::doButtonAcceptROICallback ( Widget    w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doButtonAcceptROI ( w, callData );
}
```

```
void BbRROIUI::doButtonRemoveCallback ( Widget    w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doButtonRemove ( w, callData );
}
```

```
void BbRROIUI::doButtonSaveROICallback ( Widget    w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doButtonSaveROI ( w, callData );
}
```

```

}

void BbRROIUI::doOptionBackFlowCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionBackFlow ( w, callData );
}

void BbRROIUI::doOptionHideCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionHide ( w, callData );
}

void BbRROIUI::doOptionHideNeighborCallback ( Widget      w,
                                              XtPointer clientData,
                                              XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionHideNeighbor ( w, callData );
}

void BbRROIUI::doOptionModifyCallback ( Widget      w,
                                        XtPointer clientData,
                                        XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionModify ( w, callData );
}

void BbRROIUI::doOptionModify3DCallback ( Widget      w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionModify3D ( w, callData );
}

void BbRROIUI::doOptionOpenROICallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionOpenROI ( w, callData );
}

void BbRROIUI::doOptionROIFlowCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionROIFlow ( w, callData );
}

void BbRROIUI::doOptionSave3DCallback ( Widget      w,
                                        XtPointer clientData,
                                        XtPointer callData )
{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionSave3D ( w, callData );
}

```



```

void BbRROIUI::doOptionShowCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionShow ( w, callData );
}

void BbRROIUI::doOptionShow3DCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionShow3D ( w, callData );
}

void BbRROIUI::doOptionShowAllCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionShowAll ( w, callData );
}

void BbRROIUI::doOptionShowAllNeighborCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionShowAllNeighbor ( w, callData );
}

void BbRROIUI::doOptionShowNeighborCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->doOptionShowNeighbor ( w, callData );
}

void BbRROIUI::roisCallback ( Widget      w,
                              XtPointer clientData,
                              XtPointer callData )

{
    BbRROIUI* obj = ( BbRROIUI * ) clientData;
    obj->rois ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbRROIUI::NextNeighbor ( Widget, XtPointer )
{
    // This virtual function is called from NextNeighborCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::PrevROI ( Widget, XtPointer )
{
    // This virtual function is called from PrevROICallback.
    // This function is normally overridden by a derived class.
}

```

```

}

void BbRROIUI::ROIName ( Widget, XtPointer )
{
    // This virtual function is called from ROINameCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doButtonAcceptROI ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptROICallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doButtonRemove ( Widget, XtPointer )
{
    // This virtual function is called from doButtonRemoveCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doButtonSaveROI ( Widget, XtPointer )
{
    // This virtual function is called from doButtonSaveROICallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionBackFlow ( Widget, XtPointer )
{
    // This virtual function is called from doOptionBackFlowCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionHide ( Widget, XtPointer )
{
    // This virtual function is called from doOptionHideCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionHideNeighbor ( Widget, XtPointer )
{
    // This virtual function is called from doOptionHideNeighborCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionModify ( Widget, XtPointer )
{
    // This virtual function is called from doOptionModifyCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionModify3D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionModify3DCallback.
    // This function is normally overridden by a derived class.
}

```

```

void BbRROIUI::doOptionOpenROI ( Widget, XtPointer )
{
    // This virtual function is called from doOptionOpenROICallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionROIFlow ( Widget, XtPointer )
{
    // This virtual function is called from doOptionROIFlowCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionSave3D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionSave3DCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionShow ( Widget, XtPointer )
{
    // This virtual function is called from doOptionShowCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionShow3D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionShow3DCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionShowAll ( Widget, XtPointer )
{
    // This virtual function is called from doOptionShowAllCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionShowAllNeighbor ( Widget, XtPointer )
{
    // This virtual function is called from doOptionShowAllNeighborCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::doOptionShowNeighbor ( Widget, XtPointer )
{
    // This virtual function is called from doOptionShowNeighborCallback.
    // This function is normally overridden by a derived class.
}

void BbRROIUI::rois ( Widget, XtPointer )
{
    // This virtual function is called from roisCallback.
    // This function is normally overridden by a derived class.
}

```

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for BbRTable
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRTableUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbRTable.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbRTableUI and are
// available as protected data members inherited by this class
//
// XmList                _scrolledListVessel3
// XmLabel                _labelUnitsNum
// XmLabel                _labelUnits
// XmList                _scrolledListVessel1
// XmLabel                _labelMin1
// XmLabel                _labelMinNum1
// XmLabel                _labelMax1
// XmLabel                _labelMaxNum1
// XmLabel                _labelAverage1
// XmLabel                _labelAverageNum1
// XmLabel                _labelHeartRate1
// XmLabel                _labelVolume1
// XmLabel                _labelHeartRateNum1
// XmLabel                _labelVolumeNum1
// VkOptionMenu *        _optionMenuFlow2
// VkMenuItem *          _optionVFR2
// VkMenuItem *          _optionPSV2
// VkMenuItem *          _optionBSV2
// VkMenuItem *          _optionASV2
// VkMenuItem *          _optionArea2
//
////////////////////////////////////
///---- Start editable code block: headers and declarations

```

```
#include "Utility.h"
#include "Utility_Widget.h"
```

485

```
//---- End editable code block: headers and declarations
```

```
//---- BbRTable Constructor
```

```
BbRTable::BbRTable(const char *name, Widget parent) :
    BbRTableUI(name, parent)
```

```
{
    // This constructor calls BbRTableUI(parent, name)
    // which calls BbRTableUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built
```

```
//---- Start editable code block: BbRTable constructor
```

```
_current_vessel = 0;
```

```
//---- End editable code block: BbRTable constructor
```

```
}    // End Constructor
```

```
BbRTable::BbRTable(const char *name) :
    BbRTableUI(name)
```

```
{
    // This constructor calls BbRTableUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used
```

```
//---- Start editable code block: BbRTable constructor 2
```

```
_current_vessel = 0;
```

```
//---- End editable code block: BbRTable constructor 2
```

```
}    // End Constructor
```

```
BbRTable::~~BbRTable()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.
```

```
//---- Start editable code block: BbRTable destructor
```

```
//---- End editable code block: BbRTable destructor
```

```
}    // End Destructor
```

```

const char * BbRTable::className() // classname
{
    return ("BbRTable");
} // End className()

```

486

```

void BbRTable::doOptionASV ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRTable doOptionASV

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::doOptionASV is implemented:
    //::VkUnimplemented ( w, "BbRTable::doOptionASV" );

    _objMag -> msgsRight.flow_select = FLOW_MV;
    show_info();

    //----- End editable code block: BbRTable doOptionASV
} // End BbRTable::doOptionASV()

```

```

void BbRTable::doOptionArea ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRTable doOptionArea

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::doOptionArea is implemented:
    //::VkUnimplemented ( w, "BbRTable::doOptionArea" );

    _objMag -> msgsRight.flow_select = FLOW_AREA;
    show_info();

    //----- End editable code block: BbRTable doOptionArea
} // End BbRTable::doOptionArea()

```

```

void BbRTable::doOptionBSV ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRTable doOptionBSV

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::doOptionBSV is implemented:
    //::VkUnimplemented ( w, "BbRTable::doOptionBSV" );

    _objMag -> msgsRight.flow_select = FLOW_BSV;
    show_info();

    //----- End editable code block: BbRTable doOptionBSV
} // End BbRTable::doOptionBSV()

```

```

void BbRTable::doOptionPSV ( Widget w, XtPointer callData )
{
    //----- Start editable code block: BbRTable doOptionPSV

```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData487

//--- Comment out the following line when BbRTable::doOptionPSV is implemented:
//::VkUnimplemented ( w, "BbRTable::doOptionPSV" );

_objMag -> msgsRight.flow_select = FLOW_PSV;
show_info();

//---- End editable code block: BbRTable doOptionPSV
} // End BbRTable::doOptionPSV()

void BbRTable::doOptionVFR ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRTable doOptionVFR

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::doOptionVFR is implemented:
    //::VkUnimplemented ( w, "BbRTable::doOptionVFR" );

    _objMag -> msgsRight.flow_select = FLOW_VFR;
    show_info();

    //---- End editable code block: BbRTable doOptionVFR
} // End BbRTable::doOptionVFR()

void BbRTable::vessel ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRTable vessel

    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::vessel is implemented:
    //::VkUnimplemented ( w, "BbRTable::vessel" );

    //---- End editable code block: BbRTable vessel
} // End BbRTable::vessel()

void BbRTable::vesselRTable ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRTable vesselRTable

    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbRTable::vesselRTable is implemented:
    //::VkUnimplemented ( w, "BbRTable::vesselRTable" );

    for(int i=0; i<_objMag ->_num_vessels; i++)
        if( XmListPosSelected(w, i) ) break;

```



```

if(i==0) i = _objMag num_vessels - 1;
else --i;

_current_vessel = i;

//---- End editable code block: BbRTable vesselRTable
..} // End BbRTable::vesselRTable()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *BbRTable::CreateBbRTable( const char *name, Widget parent )
{
    VkComponent *obj = new BbRTable ( name, parent );
    return ( obj );
} // End CreateBbRTable

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

void *BbRTable::RegisterBbRTableInterface()
..{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char * (Use XmRString)
    // Boolean (Use XmRBoolean)
    // int (Use XmRInt)
    // float (Use XmRFloat)
    // No argument (Use VkRNoArg or "NoArg")
    // A filename (Use VkRFilename or "Filename")
    // An enumeration (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")

```

```
static InterfaceMap map[] = {
//---- Start editable code block: BbRTableUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbRTableUI resource table
    { NULL }, // MUST be NULL terminated
};
```

```
    return map;
} // End RegisterBbRTableInterface()
```

```
//---- End of generated code
```

```
//---- Start editable code block: End of generated code
```

```
void BbRTable::show_info()
```

```
{
    int i, j, k;
    int vessel = _current_vessel;
    int num = _objMag->_flow[vessel].numPoints;
    float *y0 = new float[num];

    for(i=0; i<num; i++)
    {
        switch (_objMag->msgsRight.flow_select)
        {
            case FLOW_VFR:
                y0[i] = _objMag->_flow[vessel].vesselFlows[i].vfr;
                break;
            case FLOW_PSV:
                y0[i] = _objMag->_flow[vessel].vesselFlows[i].psv;
                break;
            case FLOW_BSV:
                y0[i] = _objMag->_flow[vessel].vesselFlows[i].bsv;
                break;
            case FLOW_MV:
                y0[i] = _objMag->_flow[vessel].vesselFlows[i].mv;
                break;
            case FLOW_AREA:
                y0[i] = _objMag->_flow[vessel].vesselFlows[i].area;
                break;
            default:
                break;
        }
    }
}
```

```
float    minI, maxI, avg;
int tag = _objMag->get_tag(num, y0, &minI, &maxI, &avg);
avg *= tag;
switch (_objMag->msgsRight.flow_select)
{
    case FLOW_VFR:
        set_unit("mL/min");
        break;
    case FLOW_PSV:
        set_unit("cm/sec");
        break;
    case FLOW_BSV:
```

```

        set_unit("cm/");
        break;
    case FLOW_MV:
        set_unit("cm/sec");
        break;
    case FLOW_AREA:
        set_unit("cm^2");
        break;
    default:
        break;
}
set_info(minI, maxI, avg);

float bpm;
if(_objMag -> msgsRight.img_type == IMAGE_PCMRA
    && _objMag -> msgsRight.img_pcmra_type != PCMRA_MAGNITUDE)
    bpm = (float)(_objMag -> _img2 -> get_heart_rate());
else bpm = 0;

Utility_Widget *uw = new Utility_Widget();
uw -> set_label(_labelHeartRateNum1, int(bpm));

set_list(num, y0);

delete y0;
}

void BbRTable::set_list(int num, float *x)
{
    XmString item;
    char str[200];

    XmListDeleteAllItems(_scrolledListVessel1);
    for(int i=0; i<num; i++)
    {
        sprintf(str, "%d    %f", i+1, x[i]);
        item = XmStringCreateSimple(str);
        XmListAddItem(_scrolledListVessel1, item, i+1);
    }
}

void BbRTable::set_unit(char *str)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelUnitsNum, str);
    delete uw;
}

void BbRTable::set_info(float minI, float maxI, float avg)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelMinNum1, minI);
    uw -> set_label(_labelMaxNum1, maxI);
    uw -> set_label(_labelAverageNum1, avg);
    delete uw;
}

void BbRTable::add_vessel(char *str)
{
    XmString item = XmStringCreateSimple(str);
    XmListAddItem(_scrolledListVessel3, item, _objMag -> _num_vessels);
}

//---- End editable code block: End of generated code

```



```

////////////////////////////////////
//
// Source file for BbRTableUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbRTableUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbRTableUI::_defaultBbRTableUIResources[] = {
    "labelAverage1.labelString: Avg",
    "labelAverageNum1.labelString: 0.5",
    "labelHeartRate1.labelString: BPM",
    "labelHeartRateNum1.labelString: 70",
    "labelMax1.labelString: Max",
    "labelMaxNum1.labelString: 1",
    "labelMin1.labelString: Min",
    "labelMinNum1.labelString: 0",
    "labelUnits.labelString: Unit",
    "labelUnitsNum1.labelString: mL/min",
    "labelVolume1.labelString: Volume",
    "labelVolumeNum1.labelString: 10",
    "optionASV2.labelString: ASV",
    "optionArea2.labelString: Area",
    "optionBSV2.labelString: BSV",
    "optionMenuFlow2.labelString: ",
    "optionPSV2.labelString: PSV",
    "optionVFR2.labelString: VFR",
    "tabLabel: Table",
    "+labelAverage1.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+labelHeartRate1.fontList: SGI_DYNAMIC SmallPlainLabelFont",

```

```

    "+*labelMax1.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+*labelMin1.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+*labelUnits.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+*labelUnitsNum.fontList: SGI_DYNAMIC SmallPlainLabelFont",
    "+*labelVolume1.fontList: SGI_DYNAMIC SmallPlainLabelFont",

```

```

//---- Start editable code block: BbRTableUI Default Resources

```

```

//---- End editable code block: BbRTableUI Default Resources

```

```

(char*)NULL

```

```

};

```

```

BbRTableUI::BbRTableUI ( const char *name ) : VkComponent ( name )
{

```

```

    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

```

```

//---- Start editable code block: BbRTable constructor 2

```

```

//---- End editable code block: BbRTable constructor 2

```

```

}    // End Constructor

```

```

BbRTableUI::BbRTableUI ( const char *name, Widget parent ) : VkComponent ( name )
{

```

```

    //---- Start editable code block: BbRTable pre-create

```

```

//---- End editable code block: BbRTable pre-create

```

```

    // Call creation function to build the widget tree.

```

```

    create ( parent );

```

```

//---- Start editable code block: BbRTable constructor

```

```

//---- End editable code block: BbRTable constructor

```

```

}    // End Constructor

```

```

BbRTableUI::~BbRTableUI()
{

```

```

    // Base class destroys widgets

```

```

//---- Start editable code block: BbRTableUI destructor

```

```

//---- End editable code block: BbRTableUI destructor

```

```

}    // End destructor

```


[illegible]

```

_labelVolumeNum1 = XtCreateManagedWidget ( "labelVolumeNum1",
                                             XmLabelWidgetClass,
                                             _baseWidget,
                                             XmNlabelType, XmSTRING,
                                             XmNx, 199,
                                             XmNy, 115,
                                             XmNwidth, 30,
                                             XmNheight, 20,
                                             (XtPointer) NULL );

_optionMenuFlow2 = new VkOptionMenu ( _baseWidget, "optionMenuFlow2");
_optionVFR2 = _optionMenuFlow2->addAction ( "optionVFR2",
                                             &BbRTTableUI::doOptionVFRCallback,
                                             (XtPointer) this );

_optionPSV2 = _optionMenuFlow2->addAction ( "optionPSV2",
                                             &BbRTTableUI::doOptionPSVCallback,
                                             (XtPointer) this );

_optionBSV2 = _optionMenuFlow2->addAction ( "optionBSV2",
                                             &BbRTTableUI::doOptionBSVCallback,
                                             (XtPointer) this );

_optionASV2 = _optionMenuFlow2->addAction ( "optionASV2",
                                             &BbRTTableUI::doOptionASVCallback,
                                             (XtPointer) this );

_optionArea2 = _optionMenuFlow2->addAction ( "optionArea2",
                                             &BbRTTableUI::doOptionAreaCallback,
                                             (XtPointer) this );

XtVaSetValues ( _optionMenuFlow2->baseWidget(),
                XmNx, 10,
                XmNy, 109,
                XmNwidth, 93,
                XmNheight, 32,
                (XtPointer) NULL );

//---- Start editable code block: BbRTTableUI create

//---- End editable code block: BbRTTableUI create
}

const char * BbRTTableUI::className()
{
    return ("BbRTTableUI");
} // End className()

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

void BbRTTableUI::doOptionASVCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRTTableUI* obj = ( BbRTTableUI * ) clientData;
    obj->doOptionASV ( w, callData );
}

```

```

void BbRTableUI::doOptionAreaCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->doOptionArea ( w, callData );
}

void BbRTableUI::doOptionBSVCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->doOptionBSV ( w, callData );
}

void BbRTableUI::doOptionPSVCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->doOptionPSV ( w, callData );
}

void BbRTableUI::doOptionVFRCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->doOptionVFR ( w, callData );
}

void BbRTableUI::vesselCallback ( Widget      w,
                                  XtPointer clientData,
                                  XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->vessel ( w, callData );
}

void BbRTableUI::vesselRTableCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbRTableUI* obj = ( BbRTableUI * ) clientData;
    obj->vesselRTable ( w, callData );
}

// The following functions are called from the menu items
// in this window.

void BbRTableUI::doOptionASV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionASVCallback.
    // This function is normally overridden by a derived class.
}

void BbRTableUI::doOptionArea ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAreaCallback.
    // This function is normally overridden by a derived class.
}

```

```

}

void BbRTableUI::doOptionBSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionBSVCallback.
    // This function is normally overridden by a derived class.
}

void BbRTableUI::doOptionPSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPSVCallback.
    // This function is normally overridden by a derived class.
}

void BbRTableUI::doOptionVFR ( Widget, XtPointer )
{
    // This virtual function is called from doOptionVFRCallback.
    // This function is normally overridden by a derived class.
}

void BbRTableUI::vessel ( Widget, XtPointer )
{
    // This virtual function is called from vesselCallback.
    // This function is normally overridden by a derived class.
}

void BbRTableUI::vesselRTable ( Widget, XtPointer )
{
    // This virtual function is called from vesselRTableCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbRWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///----- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

```

```

#include "BbRWaveform.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

```

```
extern void VkUnimplemented ( Widget, const char * );
```

```

////////////////////////////////////
// The following non-container elements are created by BbRWaveformUI and are
// available as protected data members inherited by this class
//
// XmList                _scrolledListVessel2
// XmLabel               _labelUnit
// VkOptionMenu *        _optionMenuFlow1
// VkMenuItem *          _optionVFR1
// VkMenuItem *          _optionPSV1
// VkMenuItem *          _optionBSV1
// VkMenuItem *          _optionASV1
// VkMenuItem *          _optionArea1
// XmLabel               _labelCurrentNum
// XmLabel               _labelMaxNum
// XmLabel               _labelMinNum
//
////////////////////////////////////

```

```
///----- Start editable code block: headers and declarations
```

```

#include "Utility.h"
#include "Utility_Widget.h"

```

```
///----- End editable code block: headers and declarations
```

```

BbRWaveform::BbRWaveform(const char *name, Widget parent) :
    BbRWaveformUI(name, parent)
{
    // This constructor calls BbRWaveformUI(parent, name)
    // which calls BbRWaveformUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbRWaveform constructor

    _current_vessel = 0;

    //---- End editable code block: BbRWaveform constructor

} // End Constructor

```

```

BbRWaveform::BbRWaveform(const char *name) :
    BbRWaveformUI(name)
{
    // This constructor calls BbRWaveformUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbRWaveform constructor 2

    _current_vessel = 0;

    //---- End editable code block: BbRWaveform constructor 2

} // End Constructor

```

```

BbRWaveform::~BbRWaveform()
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.

    //---- Start editable code block: BbRWaveform destructor

    //---- End editable code block: BbRWaveform destructor

} // End Destructor

```

```

const char * BbRWaveform::className() // classname
{
    return ("BbRWaveform");
} // End className()

```

```

void BbRWaveform::doOptionASV ( Widget w, XtPointer callData )
{

```

```

//---- Start editable code block: BbRWaveform doOptionASV
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbRWaveform::doOptionASV is implemented:
//::VkUnimplemented ( w, "BbRWaveform::doOptionASV" );

_objMag -> msgsRight.flow_select = FLOW_MV;
_objMag -> update_Rwave(_current_vessel);

//---- End editable code block: BbRWaveform doOptionASV
} // End BbRWaveform::doOptionASV()

void BbRWaveform::doOptionArea ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRWaveform doOptionArea
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbRWaveform::doOptionArea is implemented:
//::VkUnimplemented ( w, "BbRWaveform::doOptionArea" );

_objMag -> msgsRight.flow_select = FLOW_AREA;
_objMag -> update_Rwave(_current_vessel);

//---- End editable code block: BbRWaveform doOptionArea
} // End BbRWaveform::doOptionArea()

void BbRWaveform::doOptionBSV ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRWaveform doOptionBSV
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbRWaveform::doOptionBSV is implemented:
//::VkUnimplemented ( w, "BbRWaveform::doOptionBSV" );

_objMag -> msgsRight.flow_select = FLOW_BSV;
_objMag -> update_Rwave(_current_vessel);

//---- End editable code block: BbRWaveform doOptionBSV
} // End BbRWaveform::doOptionBSV()

void BbRWaveform::doOptionPSV ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbRWaveform doOptionPSV
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

//--- Comment out the following line when BbRWaveform::doOptionPSV is implemented:
//::VkUnimplemented ( w, "BbRWaveform::doOptionPSV" );

```

```
_objMag -> msgsRight.flow_select = FLOW_PSV;
_objMag -> update_Rwave(_current_vessel);
```

```
//---- End editable code block: BbRWaveform doOptionPSV
} // End BbRWaveform::doOptionPSV()

void BbRWaveform::doOptionVFR ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRWaveform doOptionVFR
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbRWaveform::doOptionVFR is implemented:
    //::VkUnimplemented ( w, "BbRWaveform::doOptionVFR" );

    _objMag -> msgsRight.flow_select = FLOW_VFR;
    _objMag -> update_Rwave(_current_vessel);

    //---- End editable code block: BbRWaveform doOptionVFR
} // End BbRWaveform::doOptionVFR()

void BbRWaveform::vesselRWaveform ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbRWaveform vesselRWaveform
    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;

    //--- Comment out the following line when BbRWaveform::vesselRWaveform is implement
    //::VkUnimplemented ( w, "BbRWaveform::vesselRWaveform" );

    for(int i=0; i<_objMag->_num_vessels; i++)
        if( XmListPosSelected(w, i) ) break;

    if(i==0) i = _objMag->_num_vessels - 1;
    else --i;

    _current_vessel = i;

    //---- End editable code block: BbRWaveform vesselRWaveform
} // End BbRWaveform::vesselRWaveform()
```

```
////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////
```

```
VkComponent *BbRWaveform::CreateBbRWaveform( const char *name, Widget parent )
{
```



```

    VkComponent *obj = new BbRWaveform ( name, parent );
    return ( obj );
} // End CreateBbRWaveform

```

```

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

```

```

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbRWaveform::RegisterBbRWaveformInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    //     void memberFunction ( Type );
    //
    // where "Type" is one of:
    //     const char *      (Use XmRString)
    //     Boolean           (Use XmRBoolean)
    //     int               (Use XmRInt)
    //     float            (Use XmRFloat)
    //     No argument      (Use VkRNoArg or "NoArg")
    //     A filename       (Use VkRFilename or "Filename")
    //     An enumeration   (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    //     A callback       (Use XmRCallback)

```

```

static InterfaceMap map[] = {
    //---- Start editable code block: BbRWaveformUI resource table

    // { "resourceName", "setAttribute", XmRString},
    //---- End editable code block: BbRWaveformUI resource table
    { NULL }, // MUST be NULL terminated
};

```

```

    return map;
} // End RegisterBbRWaveformInterface()

```

```

//---- End of generated code

```

```

//---- Start editable code block: End of generated code

```

```
void BbRWaveform::set_unit(char *str)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelUnit, str);
    delete uw;
}

void BbRWaveform::set_info(float minI, float maxI, float avg)
{
    Utility_Widget *uw = new Utility_Widget();
    uw -> set_label(_labelMinNum, minI);
    uw -> set_label(_labelMaxNum, maxI);
    uw -> set_label(_labelCurrentNum, avg);
    delete uw;
}

void BbRWaveform::add_vessel(char *str)
{
    XmString item = XmStringCreateSimple(str);
    XmListAddItem(_scrolledListVessel2, item, _objMag -> _num_vessels);
}

//----- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for BbRWaveformUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbRWaveformUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/ScrolledW.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbRWaveformUI::_defaultBbRWaveformUIResources[] = {
    "labelCurrentNum.labelString: 0",
    "labelMaxNum.labelString: 1",
    "labelMinNum.labelString: 0",
    "labelUnit.labelString: Unit",
    "optionASV1.labelString: ASV",
    "optionArea1.labelString: Area",
    "optionBSV1.labelString: BSV",
    "optionMenuFlow1.labelString: ",
    "optionPSV1.labelString: PSV",
    "optionVFR1.labelString: VFR",
    "tabLabel: Waveform",

    //---- Start editable code block: BbRWaveformUI Default Resources

    //---- End editable code block: BbRWaveformUI Default Resources

    (char*)NULL
};

BbRWaveformUI::BbRWaveformUI ( const char *name ) : VkComponent ( name )

```

```

{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: BbRWaveform constructor 2

    //---- End editable code block: BbRWaveform constructor 2

} // End Constructor

BbRWaveformUI::BbRWaveformUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //---- Start editable code block: BbRWaveform pre-create

    //---- End editable code block: BbRWaveform pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: BbRWaveform constructor

    //---- End editable code block: BbRWaveform constructor

} // End Constructor

BbRWaveformUI::~BbRWaveformUI()
{
    // Base class destroys widgets

    //---- Start editable code block: BbRWaveformUI destructor

    //---- End editable code block: BbRWaveformUI destructor
} // End destructor

void BbRWaveformUI::create ( Widget parent )
{
    Arg      args[7];
    Cardinal count;
    count = 0;

    // Load any class-defaulted resources for this object
    setDefaultResources ( parent, _defaultBbRWaveformUIResources );

    // Create an unmanaged widget as the top of the widget hierarchy
    _baseWidget = _bbRWaveform = XtVaCreateWidget ( _name,

```

```
// install a callback to guard against unexpected widget destruction
```

```
installDestroyHandler();
```

```
// Create widgets used in this component
// All variables are data members of this class
```

[illegible][illegible]

```
XtAddCallback ( _scrolledListVessel2,
                XmNbrowseSelectionCallback,
                &BbRWaveformUI::vesselRWaveformCallback,
                (XtPointer) this );
```

```
_labelUnit = XtVaCreateManagedWidget ( "labelUnit",
                                         xmLabelWidgetClass,
                                         _baseWidget,
                                         XmNlabelType, XmSTRING,
                                         XmNx, 112,
                                         XmNy, 120,
                                         XmNwidth, 33,
                                         XmNheight, 20,
                                         (XtPointer) NULL );
```

[illegible][illegible][illegible][illegible]

```

_optionArea1 = _optionMenuFlow1->addAction ( "optionArea1",
&BbRWaveformUI::doOptionAreaCallback,
(XtPointer) this );

```

```

_labelCurrentNum = XtVaCreateManagedWidget ( "labelCurrentNum",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 158,
XmNy, 64,
XmNwidth, 20,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelMaxNum = XtVaCreateManagedWidget ( "labelMaxNum",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 157,
XmNy, 11,
XmNwidth, 20,
XmNheight, 20,
(XtPointer) NULL );

```

```

_labelMinNum = XtVaCreateManagedWidget ( "labelMinNum",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 158,
XmNy, 121,
XmNwidth, 20,
XmNheight, 20,
(XtPointer) NULL );

```

```

XtVaSetValues ( _optionMenuFlow1->baseWidget(),
XmNx, 10,
XmNy, 112,
XmNwidth, 93,
XmNheight, 32,
(XtPointer) NULL );

```

```

//---- Start editable code block: BbRWaveformUI create

```

```

//---- End editable code block: BbRWaveformUI create

```

```

}

```

```

const char * BbRWaveformUI::className()
{
    return ("BbRWaveformUI");
}
// End className()

```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void BbRWaveformUI::doOptionASVCallback ( Widget w,
XtPointer clientData,
XtPointer callData )

```

```

{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->doOptionASV ( w, callData );
}

void BbRWaveformUI::doOptionAreaCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->doOptionArea ( w, callData );
}

void BbRWaveformUI::doOptionBSVCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->doOptionBSV ( w, callData );
}

void BbRWaveformUI::doOptionPSVCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->doOptionPSV ( w, callData );
}

void BbRWaveformUI::doOptionVFRCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->doOptionVFR ( w, callData );
}

void BbRWaveformUI::vesselRWaveformCallback ( Widget      w,
                                           XtPointer clientData,
                                           XtPointer callData )
{
    BbRWaveformUI* obj = ( BbRWaveformUI * ) clientData;
    obj->vesselRWaveform ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbRWaveformUI::doOptionASV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionASVCallback.
    // This function is normally overridden by a derived class.
}

void BbRWaveformUI::doOptionArea ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAreaCallback.
    // This function is normally overridden by a derived class.
}

```

```
void BbRWaveformUI::doOptionBSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionBSVCallback.
    // This function is normally overridden by a derived class.
}

void BbRWaveformUI::doOptionPSV ( Widget, XtPointer )
{
    // This virtual function is called from doOptionPSVCallback.
    // This function is normally overridden by a derived class.
}

void BbRWaveformUI::doOptionVFR ( Widget, XtPointer )
{
    // This virtual function is called from doOptionVFRCallback.
    // This function is normally overridden by a derived class.
}

void BbRWaveformUI::vesselRWaveform ( Widget, XtPointer )
{
    // This virtual function is called from vesselRWaveformCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```



```

////////////////////////////////////
//
// Source file for BbUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbUI.h" // Generated header file for this class

#include <Xm/ArrowB.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/Separator.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

// Externally defined classes referenced by this class:

#include "DeckLTabbedDeck.h"
#include "DeckRTabbedDeck.h"
///--- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "ReadConfig.h"
#include <Vk/VkFormat.h>
#include "Utility.h"
#include "Utility_Widget.h"
#include <stdio.h>
#include "BbDisplay.h"
#include "BbLROI.h"
#include "BbAnimation.h"
#include "BbDetail.h"
#include "BbLConfig.h"

#include "BbLWaveform.h"
#include "BbRWaveform.h"
#include "BbRTable.h"
#include "BbVisual.h"
#include "BbRROI.h"

///--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

```

```

String BbUI::_defaultBbUIResources[] = {
    "*labelImgNumber.labelString: 1",
    "*option3D.labelString: 3D",
    "*optionAnimate.labelString: Animate",
    "*optionColor2D.labelString: 2D Color",
    "*optionGray2D.labelString: 2D Gray",
    "*optionMagnitude.labelString: Magnitude",
    "*optionMenuAnimate.labelString: ",
    "*optionMenuPCMRA.labelString: ",
    "*optionMenuSelect.labelString: ",
    "*optionMenuSpace.labelString: ",
    "*optionMenuVisual.labelString: ",
    "*optionNewAnimate.labelString: New",
    "*optionOther.labelString: Other",
    "*optionPhase.labelString: Phase",
    "*optionROI.labelString: ROI",
    "*optionReference.labelString: Reference",
    "*optionSimple.labelString: Simple",
    "*optionSpline.labelString: Spline",
    "*optionStopAnimate.labelString: Stop",
    "*optionVelocity.labelString: Velocity",
    "*optionWhole.labelString: Whole",
    "+*labelImgNumber.fontList: SGI_DYNAMIC SmallPlainLabelFont",

    //---- Start editable code block: BbUI Default Resources

    //---- End editable code block: BbUI Default Resources

    (char*)NULL
};

BbUI::BbUI ( const char *name ) : VkComponent ( name )
{
    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

    //---- Start editable code block: Bb constructor 2

    //---- End editable code block: Bb constructor 2

}    // End Constructor

BbUI::BbUI ( const char *name, Widget parent ) : VkComponent ( name )
{
    //---- Start editable code block: Bb pre-create

    //---- End editable code block: Bb pre-create

    // Call creation function to build the widget tree.
    create ( parent );

    //---- Start editable code block: Bb constructor

```

[illegible]

```
_optionStopAnimate = _optionMenuAnimate->addAction ( "optionStopAnimate",
                                                    &BbUI::doOptionStopAnimateCal
                                                    (XtPointer) this );

_optionNewAnimate = _optionMenuAnimate->addAction ( "optionNewAnimate",
                                                    &BbUI::doOptionNewAnimateCallk
                                                    (XtPointer) this );

_optionMenuSpace = new VkOptionMenu ( _baseWidget, "optionMenuSpace");
_optionGray2D = _optionMenuSpace->addAction ( "optionGray2D",
                                              &BbUI::doOptionGray2DCallback,
                                              (XtPointer) this );

_optionColor2D = _optionMenuSpace->addAction ( "optionColor2D",
                                              &BbUI::doOptionColor2DCallback,
                                              (XtPointer) this );

_option3D = _optionMenuSpace->addAction ( "option3D",
                                          &BbUI::doOption3DCallback,
                                          (XtPointer) this );

_optionMenuVisual = new VkOptionMenu ( _baseWidget, "optionMenuVisual");
_optionSpline = _optionMenuVisual->addAction ( "optionSpline",
                                              &BbUI::doOptionSplineCallback,
                                              (XtPointer) this );

_optionSimple = _optionMenuVisual->addAction ( "optionSimple",
                                              &BbUI::doOptionSimpleCallback,
                                              (XtPointer) this );

_optionMenuSelect = new VkOptionMenu ( _baseWidget, "optionMenuSelect");
_optionWhole = _optionMenuSelect->addAction ( "optionWhole",
                                              &BbUI::doOptionWholeCallback,
                                              (XtPointer) this );

_optionROI = _optionMenuSelect->addAction ( "optionROI",
                                           &BbUI::doOptionROICallback,
                                           (XtPointer) this );

_optionReference = _optionMenuSelect->addAction ( "optionReference",
                                                  &BbUI::doOptionReferenceCallback
                                                  (XtPointer) this );

_optionOther = _optionMenuSelect->addAction ( "optionOther",
                                              &BbUI::doOptionOtherCallback,
                                              (XtPointer) this );

_arrowNext = XtVaCreateManagedWidget ( "arrowNext",
                                       xmArrowButtonWidgetClass,
                                       _baseWidget,
                                       XmNarrowDirection, XmARROW_UP,
                                       XmNx, 490,
                                       XmNy, 630,
                                       XmNwidth, 50,
                                       XmNheight, 50,
                                       (XtPointer) NULL );

XtAddCallback ( _arrowNext,
               XmNactivateCallback,
               &BbUI::NextCallback,
               (XtPointer) this );

_arrowPrev = XtVaCreateManagedWidget ( "arrowPrev",
```

```
xmArrowButtonWidgetClass,
_baseWidget,
XmNarrowDirection, XmARROW_DOWN,
XmNx, 420,
XmNy, 630,
XmNwidth, 50,
XmNheight, 50,
(XtPointer) NULL );
```

516



```
XtAddCallback ( _arrowPrev,
                XmNactivateCallback,
                &BbUI::PrevCallback,
                (XtPointer) this );
```

```
_labelImgNumber = XtVaCreateManagedWidget ( "labelImgNumber",
                                              xmLabelWidgetClass,
                                              _baseWidget,
                                              XmNalignment, XmALIGNMENT_BEGINNING,
                                              XmNlabelType, XmSTRING,
                                              XmNx, 80,
                                              XmNy, 650,
                                              XmNwidth, 11,
                                              XmNheight, 18,
                                              (XtPointer) NULL );
```

```
_separatorTop = XtVaCreateManagedWidget ( "separatorTop",
                                           xmSeparatorWidgetClass,
                                           _baseWidget,
                                           XmNorientation, XmHORIZONTAL,
                                           XmNx, 12,
                                           XmNy, 30,
                                           XmNwidth, 1220,
                                           XmNheight, 20,
                                           (XtPointer) NULL );
```

```
_separatorBottom = XtVaCreateManagedWidget ( "separatorBottom",
                                              xmSeparatorWidgetClass,
                                              _baseWidget,
                                              XmNorientation, XmHORIZONTAL,
                                              XmNx, 10,
                                              XmNy, 692,
                                              XmNwidth, 1220,
                                              XmNheight, 20,
                                              (XtPointer) NULL );
```

```
_separatorMiddle = XtVaCreateManagedWidget ( "separatorMiddle",
                                              xmSeparatorWidgetClass,
                                              _baseWidget,
                                              XmNorientation, XmVERTICAL,
                                              XmNx, 619,
                                              XmNy, 31,
                                              XmNwidth, 20,
                                              XmNheight, 900,
                                              (XtPointer) NULL );
```

```
_deckR = new DeckRTabbedDeck( "deckR", _baseWidget );
_deckR->show();
```

```
_deckL = new DeckLTabbedDeck( "deckL", _baseWidget );
_deckL->show();
```

```

XtVaSetValues ( _optionMenuPCMRA->baseWidget(),
                XmNx, 623,
                XmNy, 642,
                XmNwidth, 135,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuAnimate->baseWidget(),
                XmNx, 1114,
                XmNy, 655,
                XmNwidth, 117,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuSpace->baseWidget(),
                XmNx, 964,
                XmNy, 655,
                XmNwidth, 122,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuVisual->baseWidget(),
                XmNx, 819,
                XmNy, 654,
                XmNwidth, 108,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _optionMenuSelect->baseWidget(),
                XmNx, 660,
                XmNy, 653,
                XmNwidth, 131,
                XmNheight, 32,
                (XtPointer) NULL );
XtVaSetValues ( _deckR->baseWidget(),
                XmNx, 637,
                XmNy, 709,
                XmNwidth, 600,
                XmNheight, 220,
                (XtPointer) NULL );
XtVaSetValues ( _deckL->baseWidget(),
                XmNx, 10,
                XmNy, 710,
                XmNwidth, 610,
                XmNheight, 220,
                (XtPointer) NULL );

//---- Start editable code block: BbUI create

init();

//---- End editable code block: BbUI create
}

const char * BbUI::className()
{
    return ("BbUI");
} // End className()

////////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////////

void BbUI::NextCallback ( Widget w,
                        XtPointer clientData,
                        XtPointer callData )

```

```

BbUI* obj = ( BbUI * ) clientData;
obj->Next ( w, callData );
}

void BbUI::PrevCallback ( Widget      w,
                          XtPointer clientData,
                          XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->Prev ( w, callData );
}

void BbUI::doOption3DCallback ( Widget      w,
                               XtPointer clientData,
                               XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOption3D ( w, callData );
}

void BbUI::doOptionAnimateCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionAnimate ( w, callData );
}

void BbUI::doOptionColor2DCallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionColor2D ( w, callData );
}

void BbUI::doOptionGray2DCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionGray2D ( w, callData );
}

void BbUI::doOptionMagnitudeCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionMagnitude ( w, callData );
}

void BbUI::doOptionNewAnimateCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionNewAnimate ( w, callData );
}

void BbUI::doOptionOtherCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;

```

```

    obj->doOptionOther ( callData );
}

void BbUI::doOptionPhaseCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionPhase ( w, callData );
}

void BbUI::doOptionROICallback ( Widget      w,
                                 XtPointer clientData,
                                 XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionROI ( w, callData );
}

void BbUI::doOptionReferenceCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionReference ( w, callData );
}

void BbUI::doOptionSimpleCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionSimple ( w, callData );
}

void BbUI::doOptionSplineCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionSpline ( w, callData );
}

void BbUI::doOptionStopAnimateCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionStopAnimate ( w, callData );
}

void BbUI::doOptionVelocityCallback ( Widget      w,
                                      XtPointer clientData,
                                      XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionVelocity ( w, callData );
}

void BbUI::doOptionWholeCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbUI* obj = ( BbUI * ) clientData;
    obj->doOptionWhole ( w, callData );
}

```



```

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbUI::Next ( Widget, XtPointer )
{
    // This virtual function is called from NextCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::Prev ( Widget, XtPointer )
{
    // This virtual function is called from PrevCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOption3D ( Widget, XtPointer )
{
    // This virtual function is called from doOption3DCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionAnimate ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAnimateCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionColor2D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionColor2DCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionGray2D ( Widget, XtPointer )
{
    // This virtual function is called from doOptionGray2DCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionMagnitude ( Widget, XtPointer )
{
    // This virtual function is called from doOptionMagnitudeCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionNewAnimate ( Widget, XtPointer )
{
    // This virtual function is called from doOptionNewAnimateCallback.
    // This function is normally overridden by a derived class.
}

void BbUI::doOptionOther ( Widget, XtPointer )
{

```

```
// This virtual function is called from doOptionOtherCallback.  
// This function is normally overridden by a derived class.
```

521

```
}
```

```
void BbUI::doOptionPhase ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionPhaseCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionROI ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionROICallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionReference ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionReferenceCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionSimple ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionSimpleCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionSpline ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionSplineCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionStopAnimate ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionStopAnimateCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionVelocity ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionVelocityCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
void BbUI::doOptionWhole ( Widget, XtPointer )
```

```
{
```

```
    // This virtual function is called from doOptionWholeCallback.  
    // This function is normally overridden by a derived class.
```

```
}
```

```
//---- Start editable code block: End of generated code
```

```
void BbUI::set(ObjectManager *objMag)
```

```

{
    _objMag = objMag;
    objMag -> set(this);

    _deckR -> set(objMag);
    _deckL -> set(objMag);
}

void BbUI::init_patient()
{
    Utility *u = new Utility();

    int img_type = u -> get_ImgType(_objMag->msgsLoaded.img_type);

    _objMag->msgsLeft.img_type = img_type;
    if(img_type != IMAGE_PCMRA) printf("\n\n New but Not PCMRA\n");
    else printf("\n\n NEW PCMRA\n");

    _objMag->msgsLeft.img_anatomy = u -> get_ImgAnatomy(_objMag->msgsLeft.img_type,
        _objMag->msgsLoaded.img_anatomy);

    _objMag->msgsRight.num_imgs = _objMag->msgsLoaded.img_end - _objMag->msgsLoaded.i

    //remove_flow();
    //_objMag->_num_vessels = 0;
    //_objMag->_vessel = 0;
    //_objMag->_flow = init_flow(_objMag->msgsRight.num_imgs);

    /*
    if(_objMag->_root != NULL)
    {
        _objMag->_root -> unref();
        _objMag->_root = NULL;
    }
    */
    _objMag->msgsRight.img_number_prev = -1;

    delete _objMag-> _ROIS;
    _objMag-> _ROIS = new ROIS(_objMag->msgsRight.num_imgs);

    _objMag->msgsRight.img_pcmra_type = PCMRA_MAGNITUDE;
    _objMag->msgsRight.ratio3D = 0;

    _objMag -> get_general();

    _objMag -> update_Aimj(_objMag->msgsLoaded.img_start);

    Utility_Widget *uw = new Utility_Widget();
    uw->set_label(((BbDisplay *) (_objMag->_LDisp))->_labelDisplayTotalNum, _objMag->n

    delete u;
    delete uw;
}

void BbUI::init()
{
    //
    // Create ObjectManager
    //
    ObjectManager *objMag = new ObjectManager();

    //
    // Get objMag->msgsL by Reading a file
    //
    objMag->msgsLoaded = ReadConfig();
}

```

```
int img_type = u -> get_ImgType(objMag->msgsLoaded.img_type);
```

```
objMag->msgsLeft = ReadConfigLeft(img_type);
```

```
objMag->msgsRight = ReadConfigRight(img_type);
```

```
objMag->msgsLeft.img_type = img_type;
```

```
if(img_type != IMAGE_PCMRA) printf(" Not PCMRA\n");
```

```
objMag->msgsLeft.img_anatomy = u -> get_ImgAnatomy(objMag->msgsLeft.img_type,
    objMag->msgsLoaded.img_anatomy);
```

```
objMag -> init();
```

```
/*
```

```
float winCenter, winWidth;
```

```
u -> get_GE(objMag->msgsLeft.img_type, objMag->msgsLeft.img_anatomy,
    &winCenter, &winWidth);
```

```
objMag->msgsLeft.img_winCenter = winCenter;
```

```
objMag->msgsLeft.img_winWidth = winWidth;
```

```
objMag->msgsRight.img_winCenter = winCenter;
```

```
objMag->msgsRight.img_winWidth = winWidth;
```

```
*/
```

```
//
```

```
// Set current image number
```

```
//
```

```
objMag->msgsRight.num_imgs = objMag->msgsLoaded.img_end - objMag->msgsLoaded.img_
```

```
//objMag->msgsRight.flows = new FlowPara[objMag->msgsRight.num_imgs];
```

```
objMag->_num_vessels = 0;
```

```
objMag->_vessel = 0;
```

```
objMag->_flow = init_flow(objMag->msgsRight.num_imgs);
```

```
//
```

```
// Initialize ROIS
```

```
//
```

```
objMag-> _ROIS = new ROIS(objMag->msgsRight.num_imgs);
```

```
//objMag-> _ROIS -> from_File("ROI.DAT");
```

```
//
```

```
// Set a Link to BbUI, DeckLTabbedDeck, DeckRTabbedDeck
```

```
//
```

```
set(objMag);
```

```
delete u;
```

```
if(objMag->msgsLeft.img_type == IMAGE_PCMRA)
```

```
{
```

```
    objMag->msgsRight.flowDir = -1;
```

```
    objMag->msgsRight.flowDir2 = -1;
```

```
}
```

```
else
```

```
    objMag->msgsRight.flowDir = 0;
```

```
objMag -> get_general();
```

```
objMag -> update_Aimg(objMag->msgsLoaded.img_start);
```

```
(((BbRROI *) (objMag -> _RROI)) -> set_list());
```

```
//GE_PCMRA_HEADER_OBJ *pc = objMag -> _img -> get_header();
```

```
//objMag-> _ROIS -> to_ivFile(pc->slthick, pc->pixsize_X, pc->pixsize_Y);
```

```
//
```

```

// Set _labelDisplayTotalNum
//
Utility_Widget *u1 = new Utility_Widget();
u1->set_label(((BbDisplay *) (objMag->_LDisp))->_labelDisplayTotalNum, objMag->msg);
delete u1;
printf("BbUI:: init is done \n");
//getchar();

((BbLROI *) (objMag -> _LROI)) -> init2();
((BbDisplay *) (objMag->_LDisp))-> init();
((BbDetail *) (objMag->_LDet1))-> init();
((BbAnimation *) (objMag->_RAnimate))-> init();

objMag->_patients = ((BbLConfig *) (objMag->_LConfig)) -> init();
}

Flow *BbUI::init_flow(int num)
{
    int i, j;

    Flow *flow = new Flow[50];
    for(i=0; i<50; i++)
    {
        flow[i].vesselFlows = new FlowPara[32];
        for(j=0; j<num; j++)
        {
            flow[i].vesselFlows[j].vfr = 0;
            flow[i].vesselFlows[j].psv = 0;
            flow[i].vesselFlows[j].bsv = 0;
            flow[i].vesselFlows[j].mv = 0;
            flow[i].vesselFlows[j].area = 0;
        }
    }
    return flow;
}

void BbUI::remove_flow()
{
    for(int i=0; i<50; i++)
    {
        delete _objMag -> _flow[i].vesselFlows;
    }
    delete _objMag -> _flow;

    ((BbVisual *) (_objMag -> _RVis1)) -> clear_vessel();
    ((BbLWaveform *) (_objMag -> _LWave)) -> clear_vessel();
    ((BbRWaveform *) (_objMag -> _RWave)) -> clear_vessel();
    ((BbRTable *) (_objMag -> _RTable)) -> clear_vessel();
}

//----- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbVelocity
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVelocityUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbVelocity.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbVelocityUI and are
// available as protected data members inherited by this class
//
// XmTextField          _textfieldMagThresh
// XmLabel              _labelMagThresh
// XmLabel              _labelNegThresh
// XmLabel              _labelPosThresh
// XmLabel              _labelPhase2Velocity
// XmTextField          _textfieldNegThresh
// XmTextField          _textfieldPosThresh
// VkOptionMenu *       _optionMenu4
// VkMenuItem *         _optionNone
// VkMenuItem *         _option25
// VkMenuItem *         _option1
// VkMenuItem *         _option75
// VkMenuItem *         _option100
// XmTextField          _textfieldRatio
// VkOptionMenu *       _optionMenuVelocityMethod
// VkMenuItem *         _optionAsIs
// VkMenuItem *         _optionROIMasked
// VkMenuItem *         _optionFlowMasked
//
////////////////////////////////////

//---- Start editable code block: headers and declarations

#include "Utility.h"

```

//---- BbVelocity Constructor

```
BbVelocity::BbVelocity(const char *name, Widget parent) :
    BbVelocityUI(name, parent)
```

```
{
    // This constructor calls BbVelocityUI(parent, name)
    // which calls BbVelocityUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built
```

//---- Start editable code block: BbVelocity constructor

//---- End editable code block: BbVelocity constructor

```
}    // End Constructor
```

```
BbVelocity::BbVelocity(const char *name) :
    BbVelocityUI(name)
```

```
{
    // This constructor calls BbVelocityUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used
```

//---- Start editable code block: BbVelocity constructor 2

//---- End editable code block: BbVelocity constructor 2

```
}    // End Constructor
```

```
BbVelocity::~BbVelocity()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.
```

//---- Start editable code block: BbVelocity destructor

//---- End editable code block: BbVelocity destructor

```
}    // End Destructor
```

```
const char * BbVelocity::className() // classname
```

```
{
    return ("BbVelocity");
} // End className()
```

```

void BbVelocity::Ratio ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity Ratio

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::Ratio is implemented:
    //::VkUnimplemented ( w, "BbVelocity::Ratio" );

    //---- End editable code block: BbVelocity Ratio
}    // End BbVelocity::Ratio()

void BbVelocity::doOption100 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOption100

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::doOption100 is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOption100" );

    XmTextFieldSetString(_textfieldRatio, "100.0");

    //---- End editable code block: BbVelocity doOption100
}    // End BbVelocity::doOption100()

void BbVelocity::doOption25 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOption25

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::doOption25 is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOption25" );

    XmTextFieldSetString(_textfieldRatio, "25.0");

    //---- End editable code block: BbVelocity doOption25
}    // End BbVelocity::doOption25()

void BbVelocity::doOption50 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOption50

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::doOption50 is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOption50" );

    XmTextFieldSetString(_textfieldRatio, "50.0");
}

```



```

//---- End editable code block: BbVelocity doOption50
} // End BbVelocity::doOption50()

void BbVelocity::doOption75 ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOption75
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::doOption75 is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOption75" );

    XmTextFieldSetString(_textfieldRatio, "75.0");
    //---- End editable code block: BbVelocity doOption75
} // End BbVelocity::doOption75()

void BbVelocity::doOptionAsIs ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOptionAsIs
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::doOptionAsIs is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOptionAsIs" );

    _objMag -> msgsRight.velocity_select = VELOCITY_ASIS;
    if(_objMag -> msgsRight.img_select == RIGHT_IMG_ROI &&
        _objMag -> msgsRight.img_pcmra_type == PCMRA_VELOCITY)
        _objMag -> update_Rimg(_objMag -> msgsRight.img_number);

    //---- End editable code block: BbVelocity doOptionAsIs
} // End BbVelocity::doOptionAsIs()

void BbVelocity::doOptionFlowMasked ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOptionFlowMasked
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::doOptionFlowMasked is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOptionFlowMasked" );

    _objMag -> msgsRight.velocity_select = VELOCITY_FLOWMASKED;
    if(_objMag -> msgsRight.img_select == RIGHT_IMG_ROI &&
        _objMag -> msgsRight.img_pcmra_type == PCMRA_VELOCITY)
    {
        _objMag -> msgsRight.velocity_ratio = atof(XmTextFieldGetString(_textfieldRatio));
        printf(" BbVelocity %f \n", _objMag -> msgsRight.velocity_ratio);
        _objMag -> update_Rimg(_objMag -> msgsRight.img_number);
    }

    //---- End editable code block: BbVelocity doOptionFlowMasked
}

```

```

void BbVelocity::doOptionNone ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOptionNone
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::doOptionNone is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOptionNone" );
    XmTextFieldSetString(_textfieldRatio, "10.0");

    //---- End editable code block: BbVelocity doOptionNone
} // End BbVelocity::doOptionNone()

void BbVelocity::doOptionROIMasked ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity doOptionROIMasked
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::doOptionROIMasked is implemented:
    //::VkUnimplemented ( w, "BbVelocity::doOptionROIMasked" );
    _objMag -> msgsRight.velocity_select = VELOCITY_ROIMASKED;
    if(_objMag -> msgsRight.img_select == RIGHT_IMG_ROI &&
        _objMag -> msgsRight.img_pcmra_type == PCMRA_VELOCITY)
    {
        _objMag -> msgsRight.velocity_ratio = atof(XmTextFieldGetString(_textfieldRatio))
        printf(" BbVelocity %f \n", _objMag -> msgsRight.velocity_ratio);
        _objMag -> update_Rimg(_objMag -> msgsRight.img_number);
    }
    //---- End editable code block: BbVelocity doOptionROIMasked
} // End BbVelocity::doOptionROIMasked()

void BbVelocity::threshMag ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity threshMag
    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;
    //--- Comment out the following line when BbVelocity::threshMag is implemented:
    //::VkUnimplemented ( w, "BbVelocity::threshMag" );
    _objMag -> msgsLeft.magThresh = atof(XmTextFieldGetString(_textfieldMagThresh));

    //---- End editable code block: BbVelocity threshMag
} // End BbVelocity::threshMag()

void BbVelocity::threshNeg ( Widget w, XtPointer callData )

```

```

{
    //---- Start editable code block: BbVelocity threshNeg
    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::threshNeg is implemented:
    //::VkUnimplemented ( w, "BbVelocity::threshNeg" );

    _objMag -> msgsLeft.negThresh = atof(XmTextFieldGetString(_textfieldNegThresh));

    //---- End editable code block: BbVelocity threshNeg
} // End BbVelocity::threshNeg()

void BbVelocity::threshPos ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVelocity threshPos
    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbVelocity::threshPos is implemented:
    //::VkUnimplemented ( w, "BbVelocity::threshPos" );

    _objMag -> msgsLeft.posThresh = atof(XmTextFieldGetString(_textfieldPosThresh));

    //---- End editable code block: BbVelocity threshPos
} // End BbVelocity::threshPos()

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

VkComponent *BbVelocity::CreateBbVelocity( const char *name, Widget parent )
{
    VkComponent *obj = new BbVelocity ( name, parent );
    return ( obj );
} // End CreateBbVelocity

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set

```

```

void *BbVelocity::RegisterBbVelocityInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char *      (Use XmRString)
    // Boolean           (Use XmRBoolean)
    // int               (Use XmRInt)
    // float             (Use XmRFloat)
    // No argument       (Use VkrNoArg or "NoArg")
    // A filename        (Use VkrFilename or "Filename")
    // An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    // A callback        (Use XmRCallback)

    static InterfaceMap map[] = {
        //----- Start editable code block: BbVelocityUI resource table

        // { "resourceName", "setAttribute", XmRString},
        //----- End editable code block: BbVelocityUI resource table
        { NULL }, // MUST be NULL terminated
    };

    return map;
} // End RegisterBbVelocityInterface()

//----- End of generated code

//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

```

User: meide
Host: phoenix
Class: phoenix
Job: BbRROIUI.C

532

```

////////////////////////////////////
//
// Source file for BbVelocityUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbVelocityUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/TextF.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>
-///----- Start editable code block: headers and declarations

//----- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbVelocityUI::_defaultBbVelocityUIResources[] = {
    "*labelMagThresh.labelString:  Mag Thresh",
    "*labelNegThresh.labelString:  Neg Thresh",
    "*labelPhase2Velocity.labelString:  Phase to Velocity",
    "*labelPosThresh.labelString:  Pos Thresh",
    "*option1.labelString:  50%",
    "*option100.labelString:  100%",
    "*option25.labelString:  25%",
    "*option75.labelString:  75%",
    "*optionAsIs.labelString:  AS IS",
    "*optionFlowMasked.labelString:  Flow-Masked",
    "*optionMenuVelocityMethod.labelString:  ",
    "*optionNone.labelString:  None",
    "*optionROIMasked.labelString:  ROI-Masked",
    "*tabLabel:  Velocity",
    "*textfieldMagThresh.value:  0.0",
    "*textfieldNegThresh.value:  0.0",
    "*textfieldPosThresh.value:  0.0",
    "*textfieldRatio.value:  50.0",

    //----- Start editable code block: BbVelocityUI Default Resources

```

(char*)NULL

};

BbVelocityUI::BbVelocityUI (const char *name) : VkComponent (name)

{

```
// No widgets are created by this constructor.
// If an application creates a component using this constructor,
// It must explicitly call create at a later time.
// This is mostly useful when adding pre-widget creation
// code to a derived class constructor.
```

//---- Start editable code block: BbVelocity constructor 2

//---- End editable code block: BbVelocity constructor 2

} // End Constructor

BbVelocityUI::BbVelocityUI (const char *name, Widget parent) : VkComponent (name)

{

//---- Start editable code block: BbVelocity pre-create

//---- End editable code block: BbVelocity pre-create

// Call creation function to build the widget tree.

create (parent);

//---- Start editable code block: BbVelocity constructor

//---- End editable code block: BbVelocity constructor

} // End Constructor

BbVelocityUI::~BbVelocityUI()

{

// Base class destroys widgets

//---- Start editable code block: BbVelocityUI destructor

//---- End editable code block: BbVelocityUI destructor

} // End destructor

void BbVelocityUI::create (Widget parent)

{

```
Arg      args[8];
Cardinal count;
count = 0;
```

// Load any class-defaulted resources for this object

[illegible]


```

XmNcols, 7,
XmNx, 460,
XmNy, 64,
XmNheight, 35,
(XtPointer) NULL );

```

536

```

XtAddCallback ( _textfieldRatio,
                XmNactivateCallback,
                &BbVelocityUI::RatioCallback,
                (XtPointer) this );

```

```

_optionMenuVelocityMethod = new VkOptionMenu ( _baseWidget, "optionMenuVelocityMeth
_optionAsIs = _optionMenuVelocityMethod->addAction ( "optionAsIs",
                                                    &BbVelocityUI::doOptionAsIsCa
                                                    (XtPointer) this );

```

```

_optionROIMasked = _optionMenuVelocityMethod->addAction ( "optionROIMasked",
                                                         &BbVelocityUI::doOptionF
                                                         (XtPointer) this );

```

```

_optionFlowMasked = _optionMenuVelocityMethod->addAction ( "optionFlowMasked",
                                                           &BbVelocityUI::doOption
                                                           (XtPointer) this );

```

```

XtVaSetValues ( _optionMenu4->baseWidget(),
                XmNx, 450,
                XmNy, 120,
                XmNwidth, 98,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

XtVaSetValues ( _optionMenuVelocityMethod->baseWidget(),
                XmNx, 426,
                XmNy, 10,
                XmNwidth, 156,
                XmNheight, 32,
                (XtPointer) NULL );

```

```

//---- Start editable code block: BbVelocityUI create

```

```

//---- End editable code block: BbVelocityUI create

```

```

const char * BbVelocityUI::className()
{
    return ("BbVelocityUI");
}
// End className()

```

```

////////////////////////////////////
// The following functions are static member functions used to
// interface with Motif.
////////////////////////////////////

```

```

void BbVelocityUI::RatioCallback ( Widget      w,
                                   XtPointer clientData,
                                   XtPointer callData )
{
    BbVelocityUI* obj = ( BbVelocityUI * ) clientData;
    obj->Ratio ( w, callData );
}

```

```

void BbVelocityUI::doOption100Callback ( Widget w,

```



```

{
    BbVelocityUI* obj = ( BbVelocityUI * ) clientData;
    obj->threshMag ( w, callData );
}

void BbVelocityUI::threshNegCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbVelocityUI* obj = ( BbVelocityUI * ) clientData;
    obj->threshNeg ( w, callData );
}

void BbVelocityUI::threshPosCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbVelocityUI* obj = ( BbVelocityUI * ) clientData;
    obj->threshPos ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbVelocityUI::Ratio ( Widget, XtPointer )
{
    // This virtual function is called from RatioCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOption100 ( Widget, XtPointer )
{
    // This virtual function is called from doOption100Callback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOption25 ( Widget, XtPointer )
{
    // This virtual function is called from doOption25Callback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOption50 ( Widget, XtPointer )
{
    // This virtual function is called from doOption50Callback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOption75 ( Widget, XtPointer )
{
    // This virtual function is called from doOption75Callback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOptionAsIs ( Widget, XtPointer )
{
    // This virtual function is called from doOptionAsIsCallback.
}

```

```

// This function is normally overridden by a derived class.
}

void BbVelocityUI::doOptionFlowMasked ( Widget, XtPointer )
{
    // This virtual function is called from doOptionFlowMaskedCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOptionNone ( Widget, XtPointer )
{
    // This virtual function is called from doOptionNoneCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::doOptionROIMasked ( Widget, XtPointer )
{
    // This virtual function is called from doOptionROIMaskedCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::threshMag ( Widget, XtPointer )
{
    // This virtual function is called from threshMagCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::threshNeg ( Widget, XtPointer )
{
    // This virtual function is called from threshNegCallback.
    // This function is normally overridden by a derived class.
}

void BbVelocityUI::threshPos ( Widget, XtPointer )
{
    // This virtual function is called from threshPosCallback.
    // This function is normally overridden by a derived class.
}

//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbVisual
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVisualUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
-// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "BbVisual.h"
#include <Vk/VkEZ.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/ScrolledW.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionMenu.h>
#include <Vk/VkMenuItem.h>

extern void VkUnimplemented ( Widget, const char * );

////////////////////////////////////
// The following non-container elements are created by BbVisualUI and are
// available as protected data members inherited by this class
//
// XmLabel                _labelVessel
// VkOptionMenu *         _optionMenu17
// VkMenuItem *           _vesNew
// XmSeparatorGadget      _separator8
// VkMenuItem *           _vesLCCA
// VkMenuItem *           _vesLVA
// VkMenuItem *           _vesRCCA
// VkMenuItem *           _vesRVA
// XmSeparatorGadget      _separator5
// VkMenuItem *           _vesLICANeck
// VkMenuItem *           _vesLECA
// VkMenuItem *           _vesRICANeck
// VkMenuItem *           _vesRECA
// XmSeparatorGadget      _separator6
// VkMenuItem *           _vesBADown
// VkMenuItem *           _vesBAup
// VkMenuItem *           _vesLICAIntra
// VkMenuItem *           _vesRICAINtra
// XmSeparatorGadget      _separator7
// VkMenuItem *           _vesLMCA
// VkMenuItem *           _vesRMCA

```

```

// VkMenuItem *           _vesLACA
// VkMenuItem *           _vesRACA
// XmTextField             _textfieldRemark
// XmLabel                 _labelDescription
// XmTextField             _textfieldDate
// XmLabel                 _labelDate
// XmList                  _scrolledListVessel4
// XmToggleButton         _toggleFlowPos
// XmToggleButton         _toggleFlowNeg
// XmToggleButton         _toggleFlowNeutral
// XmPushButton           _buttonAcceptUser
// XmLabel                  _labelFlowDir
// XmTextField             _textfieldVessel
// XmLabel                  _labelname
// XmTextField             _textfieldName
//
////////////////////////////////////

//---- Start editable code block: headers and declarations

#include "Utility.h"
#include "Utility_Widget.h"

#include "BbLWaveform.h"
#include "BbRWaveform.h"
#include "BbRTable.h"
#include "BbFormat.h"

//---- End editable code block: headers and declarations

//---- BbVisual Constructor

BbVisual::BbVisual(const char *name, Widget parent) :
    BbVisualUI(name, parent)
{
    // This constructor calls BbVisualUI(parent, name)
    // which calls BbVisualUI::create() to create
    // the widgets for this component. Any code added here
    // is called after the component's interface has been built

    //---- Start editable code block: BbVisual constructor

    //---- End editable code block: BbVisual constructor

}

// End Constructor

BbVisual::BbVisual(const char *name) :
    BbVisualUI(name)
{
    // This constructor calls BbVisualUI(name)
    // which does not create any widgets. Usually, this
    // constructor is not used

    //---- Start editable code block: BbVisual constructor 2

    //---- End editable code block: BbVisual constructor 2

```

```
} // End Constructor
```

```
BbVisual::~BbVisual()
```

```
{
    // The base class destructors are responsible for
    // destroying all widgets and objects used in this component.
    // Only additional items created directly in this class
    // need to be freed here.
```

```
//---- Start editable code block: BbVisual destructor
```

```
//---- End editable code block: BbVisual destructor
```

```
} // End Destructor
```

```
const char * BbVisual::className() // classname
```

```
{
    return ("BbVisual");
} // End className()
```

```
void BbVisual::Vessel ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual Vessel

    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::Vessel is implemented:
    //::VkUnimplemented ( w, "BbVisual::Vessel" );

    //sprintf(_objMag -> msgsRight.vesselName, "%s", XmTextFieldGetString(w));
    setVessel(XmTextFieldGetString(w));

    //---- End editable code block: BbVisual Vessel
}
```

```
    // End BbVisual::Vessel()
```

```
void BbVisual::doButtonAccept ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual doButtonAccept

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doButtonAccept is implemented:
    //::VkUnimplemented ( w, "BbVisual::doButtonAccept" );

    accept();

    //---- End editable code block: BbVisual doButtonAccept
}
```

```
    // End BbVisual::doButtonAccept()
```

```
void BbVisual::doVeLICANeck ( Widget w, XtPointer callData )
```



```

{
    //---- Start editable code block: BbVisual doVeLICANeck
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVeLICANeck is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVeLICANeck" );

    setVessel("lica-neck");

    //---- End editable code block: BbVisual doVeLICANeck
}    // End BbVisual::doVeLICANeck()

void BbVisual::doVesBAdown ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual doVesBAdown
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVesBAdown is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesBAdown" );

    setVessel("ba-down");

    //---- End editable code block: BbVisual doVesBAdown
}    // End BbVisual::doVesBAdown()

void BbVisual::doVesBAup ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual doVesBAup
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVesBAup is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesBAup" );

    setVessel("ba");

    //---- End editable code block: BbVisual doVesBAup
}    // End BbVisual::doVesBAup()

void BbVisual::doVesLACA ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual doVesLACA
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVesLACA is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesLACA" );

    setVessel("laca");

    //---- End editable code block: BbVisual doVesLACA
}

```

```
} // End BbVisual::doVesLCCA()
```

```
void BbVisual::doVesLCCA ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual doVesLCCA
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::doVesLCCA is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesLCCA" );
    setVessel("lcca");

    //---- End editable code block: BbVisual doVesLCCA
}
```

```
void BbVisual::doVesLECA ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual doVesLECA
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::doVesLECA is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesLECA" );
    setVessel("leca");

    //---- End editable code block: BbVisual doVesLECA
}
```

```
void BbVisual::doVesLICAIntra ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual doVesLICAIntra
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::doVesLICAIntra is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesLICAIntra" );
    setVessel("lica-intra");

    //---- End editable code block: BbVisual doVesLICAIntra
}
```

```
void BbVisual::doVesLMCA ( Widget w, XtPointer callData )
```

```
{
    //---- Start editable code block: BbVisual doVesLMCA
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::doVesLMCA is implemented:
```

```

//::VkUnimplemented "BbVisual::doVesLMCA" );
setVessel("lmca");

//---- End editable code block: BbVisual doVesLMCA
} // End BbVisual::doVesLMCA()

void BbVisual::doVesLVA ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesLVA
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesLVA is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesLVA" );
setVessel("lva");

//---- End editable code block: BbVisual doVesLVA
} // End BbVisual::doVesLVA()

void BbVisual::doVesNew ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesNew
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesNew is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesNew" );
setVessel("new");

//---- End editable code block: BbVisual doVesNew
} // End BbVisual::doVesNew()

void BbVisual::doVesRACA ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesRACA
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesRACA is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesRACA" );
setVessel("raca");

//---- End editable code block: BbVisual doVesRACA
} // End BbVisual::doVesRACA()

void BbVisual::doVesRCCA ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesRCCA

```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesRCCA is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesRCCA" );
setVessel("rcca");

//---- End editable code block: BbVisual doVesRCCA
} // End BbVisual::doVesRCCA()

void BbVisual::doVesRECA ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesRECA
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesRECA is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesRECA" );
setVessel("reca");

//---- End editable code block: BbVisual doVesRECA
} // End BbVisual::doVesRECA()

void BbVisual::doVesRICAItra ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesRICAItra
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesRICAItra is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesRICAItra" );
setVessel("rica-intra");

//---- End editable code block: BbVisual doVesRICAItra
} // End BbVisual::doVesRICAItra()

void BbVisual::doVesRICANeck ( Widget w, XtPointer callData )
{
//---- Start editable code block: BbVisual doVesRICANeck
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
//--- Comment out the following line when BbVisual::doVesRICANeck is implemented:
//::VkUnimplemented ( w, "BbVisual::doVesRICANeck" );
setVessel("rica-neck");

//---- End editable code block: BbVisual doVesRICANeck
} // End BbVisual::doVesRICANeck()

```

```

void BbVisual::doVesRMCA ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual doVesRMCA

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVesRMCA is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesRMCA" );

    setVessel("rmca");

    //---- End editable code block: BbVisual doVesRMCA
}    // End BbVisual::doVesRMCA()

void BbVisual::doVesRVA ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual doVesRVA

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::doVesRVA is implemented:
    //::VkUnimplemented ( w, "BbVisual::doVesRVA" );

    setVessel("rva");

    //---- End editable code block: BbVisual doVesRVA
}    // End BbVisual::doVesRVA()

void BbVisual::setToggleFlowNeg ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual setToggleFlowNeg

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::setToggleFlowNeg is implemented
    //::VkUnimplemented ( w, "BbVisual::setToggleFlowNeg" );

    set_flowdir(-1);

    //---- End editable code block: BbVisual setToggleFlowNeg
}    // End BbVisual::setToggleFlowNeg()

void BbVisual::setToggleFlowNeutral ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual setToggleFlowNeutral

    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;

    //--- Comment out the following line when BbVisual::setToggleFlowNeutral is implemented
    //::VkUnimplemented ( w, "BbVisual::setToggleFlowNeutral" );

```

```
//---- End editable code block: BbVisual setToggleFlowNeutral
} // End BbVisual::setToggleFlowNeutral()

void BbVisual::setToggleFlowPos ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual setToggleFlowPos
    XmToggleButtonCallbackStruct *cbs = (XmToggleButtonCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::setToggleFlowPos is implemented
    //::VkUnimplemented ( w, "BbVisual::setToggleFlowPos" );
    set_flowdir(1);

    //---- End editable code block: BbVisual setToggleFlowPos
} // End BbVisual::setToggleFlowPos()

void BbVisual::userName ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual userName
    XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::userName is implemented:
    //::VkUnimplemented ( w, "BbVisual::userName" );
    //sprintf(_objMag -> msgsRight.userName, "%s", XmTextFieldGetString(w));

    //---- End editable code block: BbVisual userName
} // End BbVisual::userName()

void BbVisual::vesselRUser ( Widget w, XtPointer callData )
{
    //---- Start editable code block: BbVisual vesselRUser
    XmListCallbackStruct *cbs = (XmListCallbackStruct*) callData;
    //--- Comment out the following line when BbVisual::vesselRUser is implemented:
    //::VkUnimplemented ( w, "BbVisual::vesselRUser" );
    for(int i=0; i<_objMag ->_num_vessels; i++)
        if( XmListPosSelected(w, i) ) break;
    if(i==0) i = _objMag ->_num_vessels - 1;
    else --i;
    XmTextFieldSetString(_textfieldVessel,_objMag -> _flow[i].vesselName);
    //---- End editable code block: BbVisual vesselRUser
```

```

////////////////////////////////////
// static creation function, for importing class into rapidapp
// or dynamically loading, using VkComponent::loadComponent
////////////////////////////////////

```

```

VkComponent *BbVisual::CreateBbVisual( const char *name, Widget parent )
{
    VkComponent *obj = new BbVisual ( name, parent );
    return ( obj );
} // End CreateBbVisual

```

```

////////////////////////////////////
// Function for accessing a description of the dynamic interface
// to this class.
////////////////////////////////////

```

```

// WARNING: This structure is different than that used with 1.1 RapidApp.
// See the RapidApp release notes for details

```

```

struct InterfaceMap {
    char *resourceName;
    char *methodName;
    char *argType;
    char *definingClass; // Optional, if not this class
    void (VkCallbackObject::*method)(...); // Reserved, do not set
};

```

```

void *BbVisual::RegisterBbVisualInterface()
{
    // This structure registers information about this class
    // that allows RapidApp to create and manipulate an instance.
    // Each entry provides a resource name that will appear in the
    // resource manager palette when an instance of this class is
    // selected, the name of the member function as a string,
    // the type of the single argument to this function, and an
    // optional argument indicating the class that defines this function.
    // All member functions must have the form
    //
    // void memberFunction ( Type );
    //
    // where "Type" is one of:
    // const char *      (Use XmRString)
    // Boolean           (Use XmRBoolean)
    // int               (Use XmRInt)
    // float             (Use XmRFloat)
    // No argument       (Use VkRNoArg or "NoArg")
    // A filename        (Use VkRFilename or "Filename")
    // An enumeration    (Use "Enumeration:ClassName:Type: VALUE1, VALUE2, VALUE3")
    // A callback        (Use XmRCallback)

```

```

static InterfaceMap map[] = {
//---- Start editable code block: BbVisualUI resource table

    // { "resourceName", "setAttribute", XmRString},
//---- End editable code block: BbVisualUI resource table

```

```

    { NULL }, // MUST be NULL terminated
};

return map;
} // End RegisterBbVisualInterface()

//----- End of generated code

//----- Start editable code block: End of generated code

void BbVisual::set_info(char *name, char *studyDate, char *remark)
{
    XmTextFieldSetString(_textfieldName, name);
    XmTextFieldSetString(_textfieldDate, studyDate);
    XmTextFieldSetString(_textfieldRemark, remark);
}

char *BbVisual::get_basePath()
{
    int i, i0;
    char s[300];

    sprintf(s, "%s", _objMag -> msgsLoaded.img_dir);

    int n = strlen(s);
    for(i=n-2; i>=0; i--)
    {
        if(s[i] == '/') {i0 = i; break;}
    }

    char *anatomy = new char[n-2-i0];
    char *pubPath = new char[i0];

    for(i=i0+1; i<=n-2; i++)
        anatomy[i-i0-1] = s[i];
    anatomy[n-2-i0] = '\0';
    for(i=0; i<i0; i++)
        pubPath[i] = s[i];
    pubPath[i0] = '\0';

    return pubPath;
}

void BbVisual::set_Path(char *p)
{
    FILE *fp;
    char str[300];

    if( (fp = fopen(p, "r")) == NULL )
    {
        sprintf(str, "mkdir %s", p);
        system(str);
    }
    fclose(fp);
}

void BbVisual::setVessel(char *anatomy)
{
    char str[300];
    char *basePath = new char[300];

```



```

XmTextFieldSetString(_textfieldVessel, anatomy);
basePath = get_basePath();
sprintf(str, "%s/pub", basePath);

sprintf(_objMag -> msgsRight.pubDir, "%s", str);

set_Path(str);
sprintf(str, "%s/pub/%s", basePath, anatomy);
set_Path(str);
sprintf(str, "%s/pub/%s/", basePath, anatomy);
((BbFormat *) (_objMag -> _RFormat)) -> setPath(str);
}

void BbVisual::add_flow(char *strVessel)
{
    XmString item = XmStringCreateSimple(strVessel);
    XmListAddItem(_scrolledListVessel4, item, _objMag -> _num_vessels);

    ((BbLWaveform *) (_objMag -> _LWave)) -> add_vessel( strVessel );
    ((BbRWaveform *) (_objMag -> _RWave)) -> add_vessel( strVessel );
    ((BbRTable *) (_objMag -> _RTable)) -> add_vessel( strVessel );
}

void BbVisual::set_flowdir(int dir)
{
    _objMag -> msgsRight.flowDir = dir;
    _objMag -> msgsRight.flowDir2 = _objMag -> msgsRight.flowDir;
    //Utility_Widget *uw = new Utility_Widget();
    //uw -> set_textfield(_textfieldFlowDir, dir);
    //delete uw;
}

void BbVisual::accept()
{
    char strVessel[300], strUser[300];

    sprintf(strVessel, XmTextFieldGetString(_textfieldVessel));
    sprintf(strUser, XmTextFieldGetString(_textfieldName));

    int flag = -1;
    if(_objMag -> _num_vessels > 0)
    {
        for(int i=0; i<_objMag -> _num_vessels; i++)
            if(strcmp(_objMag -> _flow[i].vesselName, strVessel) == 0)
                {flag = i; break;}
    }

    if(flag == -1)
    {
        _objMag -> _vessel = _objMag -> _num_vessels;
        ++(_objMag -> _num_vessels);
        sprintf(_objMag -> _flow[_objMag -> _vessel].vesselName, "%s", strVessel);
        sprintf(_objMag -> _flow[_objMag -> _vessel].userName, "%s", strUser);
        _objMag -> _flow[_objMag -> _vessel].numPoints = _objMag -> msgsRight.num_imgs;
        add_flow(strVessel);
    }
    else _objMag -> _vessel = flag;

    _objMag -> msgsRight.flowDir2 = _objMag -> msgsRight.flowDir;
}

//---- End editable code block: End of generated code

```

```

////////////////////////////////////
//
// Source file for BbVisualUI
//
// This class implements the user interface created in
// RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//
////////////////////////////////////

#include "BbVisualUI.h" // Generated header file for this class

#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/List.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/ScrolledW.h>
#include <Xm/TextF.h>
#include <Xm/ToggleB.h>
#include <Vk/VkResource.h>
#include <Vk/VkOptionsMenu.h>
#include <Vk/VkMenuItem.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String BbVisualUI::_defaultBbVisualUIResources[] = {
    "buttonAcceptUser.labelString: Accept",
    "labelDate.labelString: Date",
    "labelDescription.labelString: Remark",
    "labelFlowDir.labelString: Flow",
    "labelVessel.labelString: Vessel",
    "labelname.labelString: Patient",
    "tabLabel: Patient",
    "toggleFlowNeg.labelString: Negative",
    "toggleFlowNeutral.labelString: Neutral",
    "toggleFlowPos.labelString: Positive",
    "vesBAdown.labelString: BA below AICA",
    "vesBAup.labelString: BA above AICA",
    "vesLACA.labelString: LACA",
    "vesLCCA.labelString: LCCA",
    "vesLECA.labelString: LECA",
    "vesLICAIntra.labelString: LICA Intracranial",
    "vesLICANeck.labelString: LICA Neck",

```

```

"*vesLMCA.labelString: LMCA",
"*vesLVA.labelString: LVA",
"*vesNew.labelString: New",
"*vesRACA.labelString: RACA",
"*vesRCCA.labelString: RCCA",
"*vesRECA.labelString: RECA",
"*vesRICAINtra.labelString: RICA Intracranial",
"*vesRICANeck.labelString: RICA Neck",
"*vesRMCA.labelString: RMCA",
"*vesRVA.labelString: RVA",

```

```

//---- Start editable code block: BbVisualUI Default Resources

```

```

//---- End editable code block: BbVisualUI Default Resources

```

```

(char*)NULL

```

```

};

```

```

BbVisualUI::BbVisualUI ( const char *name ) : VkComponent ( name )
{

```

```

    // No widgets are created by this constructor.
    // If an application creates a component using this constructor,
    // It must explicitly call create at a later time.
    // This is mostly useful when adding pre-widget creation
    // code to a derived class constructor.

```

```

//---- Start editable code block: BbVisual constructor 2

```

```

//---- End editable code block: BbVisual constructor 2

```

```

}    // End Constructor

```

```

BbVisualUI::BbVisualUI ( const char *name, Widget parent ) : VkComponent ( name )
{

```

```

    //---- Start editable code block: BbVisual pre-create

```

```

//---- End editable code block: BbVisual pre-create

```

```

// Call creation function to build the widget tree.

```

```

    create ( parent );

```

```

//---- Start editable code block: BbVisual constructor

```

```

//---- End editable code block: BbVisual constructor

```

```

}    // End Constructor

```

```

BbVisualUI::~BbVisualUI()
{

```

```

    // Base class destroys widgets

```

```

//---- Start editable code block: BbVisualUI destructor

```



```

_separator5 = _optionMenu17->addSeparator();
_vesLICANeck = _optionMenu17->addAction ( "vesLICANeck", 555
&BbVisualUI::doVeLICANeckCallback,
(XtPointer) this );

_vesLECA = _optionMenu17->addAction ( "vesLECA",
&BbVisualUI::doVesLECACallback,
(XtPointer) this );

_vesRICANeck = _optionMenu17->addAction ( "vesRICANeck",
&BbVisualUI::doVesRICANeckCallback,
(XtPointer) this );

_vesRECA = _optionMenu17->addAction ( "vesRECA",
&BbVisualUI::doVesRECACallback,
(XtPointer) this );

_separator6 = _optionMenu17->addSeparator();
_vesBAdown = _optionMenu17->addAction ( "vesBAdown",
&BbVisualUI::doVesBAdownCallback,
(XtPointer) this );

_vesBAup = _optionMenu17->addAction ( "vesBAup",
&BbVisualUI::doVesBAupCallback,
(XtPointer) this );

_vesLICAIntra = _optionMenu17->addAction ( "vesLICAIntra",
&BbVisualUI::doVesLICAIntraCallback,
(XtPointer) this );

_vesRICAItra = _optionMenu17->addAction ( "vesRICAItra",
&BbVisualUI::doVesRICAItraCallback,
(XtPointer) this );

_separator7 = _optionMenu17->addSeparator();
_vesLMCA = _optionMenu17->addAction ( "vesLMCA",
&BbVisualUI::doVesLMCACallback,
(XtPointer) this );

_vesRMCA = _optionMenu17->addAction ( "vesRMCA",
&BbVisualUI::doVesRMCACallback,
(XtPointer) this );

_vesLACA = _optionMenu17->addAction ( "vesLACA",
&BbVisualUI::doVesLACACallback,
(XtPointer) this );

_vesRACA = _optionMenu17->addAction ( "vesRACA",
&BbVisualUI::doVesRACACallback,
(XtPointer) this );

_textfieldRemark = XtVaCreateManagedWidget ( "textfieldRemark",
xmTextFieldWidgetClass,
_baseWidget,
XmNcolumns, 20,
XmNx, 65,
XmNy, 52,
XmNheight, 35,
(XtPointer) NULL );

_labelDescription = XtVaCreateManagedWidget ( "labelDescription",
xmLabelWidgetClass,
_baseWidget,
XmNlabelType, XmSTRING,
XmNx, 10,

```



```
void BbVisualUI::VesselCallback ( Widget      w,
                                XtPointer clientData,
```


[illegible]

```

    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesLICAIntra ( w, callData );
}

void BbVisualUI::doVesLMCACallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesLMCA ( w, callData );
}

void BbVisualUI::doVesLVACallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesLVA ( w, callData );
}

void BbVisualUI::doVesNewCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesNew ( w, callData );
}

void BbVisualUI::doVesRACACallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRACA ( w, callData );
}

void BbVisualUI::doVesRCCACallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRCCA ( w, callData );
}

void BbVisualUI::doVesRECACallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRECA ( w, callData );
}

void BbVisualUI::doVesRICAItraCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRICAItra ( w, callData );
}

void BbVisualUI::doVesRICANeckCallback ( Widget      w,
                                         XtPointer clientData,
                                         XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRICANeck ( w, callData );
}

```

```

}

void BbVisualUI::doVesRMCACallback ( Widget      w,
                                     XtPointer clientData,
                                     XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRMCA ( w, callData );
}

void BbVisualUI::doVesRVACallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->doVesRVA ( w, callData );
}

void BbVisualUI::setToggleFlowNegCallback ( Widget      w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->setToggleFlowNeg ( w, callData );
}

void BbVisualUI::setToggleFlowNeutralCallback ( Widget      w,
                                                XtPointer clientData,
                                                XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->setToggleFlowNeutral ( w, callData );
}

void BbVisualUI::setToggleFlowPosCallback ( Widget      w,
                                             XtPointer clientData,
                                             XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->setToggleFlowPos ( w, callData );
}

void BbVisualUI::userNameCallback ( Widget      w,
                                    XtPointer clientData,
                                    XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->userName ( w, callData );
}

void BbVisualUI::vesselRUserCallback ( Widget      w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    BbVisualUI* obj = ( BbVisualUI * ) clientData;
    obj->vesselRUser ( w, callData );
}

////////////////////////////////////
// The following functions are called from the menu items
// in this window.
////////////////////////////////////

void BbVisualUI::Vessel ( Widget, XtPointer )

```

```

{
    // This virtual function is called from VesselCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doButtonAccept ( Widget, XtPointer )
{
    // This virtual function is called from doButtonAcceptCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVeLICANeck ( Widget, XtPointer )
{
    // This virtual function is called from doVeLICANeckCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesBAdown ( Widget, XtPointer )
{
    // This virtual function is called from doVesBAdownCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesBAup ( Widget, XtPointer )
{
    // This virtual function is called from doVesBAupCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesLACA ( Widget, XtPointer )
{
    // This virtual function is called from doVesLACACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesLCCA ( Widget, XtPointer )
{
    // This virtual function is called from doVesLCCACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesLECA ( Widget, XtPointer )
{
    // This virtual function is called from doVesLECACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesLICAIntra ( Widget, XtPointer )
{
    // This virtual function is called from doVesLICAIntraCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesLMCA ( Widget, XtPointer )
{
    // This virtual function is called from doVesLMCACallback.
    // This function is normally overridden by a derived class.
}

```

```

}

void BbVisualUI::doVesLVA ( Widget, XtPointer )
{
    // This virtual function is called from doVesLVACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesNew ( Widget, XtPointer )
{
    // This virtual function is called from doVesNewCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRACA ( Widget, XtPointer )
{
    // This virtual function is called from doVesRACACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRCCA ( Widget, XtPointer )
{
    // This virtual function is called from doVesRCCACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRECA ( Widget, XtPointer )
{
    // This virtual function is called from doVesRECACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRICAItra ( Widget, XtPointer )
{
    // This virtual function is called from doVesRICAItraCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRICANeck ( Widget, XtPointer )
{
    // This virtual function is called from doVesRICANeckCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRMCA ( Widget, XtPointer )
{
    // This virtual function is called from doVesRMCACallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::doVesRVA ( Widget, XtPointer )
{
    // This virtual function is called from doVesRVACallback.
    // This function is normally overridden by a derived class.
}
-}

```

```
void BbVisualUI::setToggleFlowNeg ( Widget, XtPointer )
{
    // This virtual function is called from setToggleFlowNegCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::setToggleFlowNeutral ( Widget, XtPointer )
{
    // This virtual function is called from setToggleFlowNeutralCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::setToggleFlowPos ( Widget, XtPointer )
{
    // This virtual function is called from setToggleFlowPosCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::userName ( Widget, XtPointer )
{
    // This virtual function is called from userNameCallback.
    // This function is normally overridden by a derived class.
}

void BbVisualUI::vesselRUser ( Widget, XtPointer )
{
    // This virtual function is called from vesselRUserCallback.
    // This function is normally overridden by a derived class.
}

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code
```

```

////////////////////////////////////
//
// Source file for DeckLTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "DeckLTabbedDeck.h"
#include <Vk/VkDeck.h>

#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include "BbDetail.h"
#include "BbDisplay.h"
#include "BbHistogram.h"
#include "BbLConfig.h"
#include "BbLConfigNew.h"
#include "BbLPCMRA.h"
#include "BbLROI.h"
#include "BbLWaveform.h"
extern void VkUnimplemented(Widget, const char *);

//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String DeckLTabbedDeck::_defaultDeckLTabbedDeckResources[] = {
    ///---- Start editable code block: DeckLTabbedDeck Default Resources

    ///---- End editable code block: DeckLTabbedDeck Default Resources

    (char*)NULL
};

//---- DeckLTabbedDeck Constructor

DeckLTabbedDeck::DeckLTabbedDeck ( const char *name, Widget parent ) :

```

```

{
    // Load any class-default resources for this object

    setDefaultResources ( baseWidget(), _defaultDeckLTabbedDeckResources );

    _bbHistogram = new BbHistogram( "bbHistogram", _vkdeck->baseWidget() );
    _bbHistogram->show();

    _bbDisplay = new BbDisplay( "bbDisplay", _vkdeck->baseWidget() );
    _bbDisplay->show();

    _bbLROI = new BbLROI( "bbLROI", _vkdeck->baseWidget() );
    _bbLROI->show();

    _bbLConfig = new BbLConfig( "bbLConfig", _vkdeck->baseWidget() );
    _bbLConfig->show();

    _bbLPCMRA = new BbLPCMRA( "bbLPCMRA", _vkdeck->baseWidget() );
    _bbLPCMRA->show();

    _bbLWaveform = new BbLWaveform( "bbLWaveform", _vkdeck->baseWidget() );
    _bbLWaveform->show();

    _bbDetail = new BbDetail( "bbDetail", _vkdeck->baseWidget() );
    _bbDetail->show();

    _bbLConfigNew = new BbLConfigNew( "bbLConfigNew", _vkdeck->baseWidget() );
    _bbLConfigNew->show();

    registerChild ( _bbHistogram, "bbHistogram");
    registerChild ( _bbDisplay, "bbDisplay");
    registerChild ( _bbLROI, "bbLROI");
    registerChild ( _bbLConfig, "bbLConfig");
    registerChild ( _bbLPCMRA, "bbLPCMRA");
    registerChild ( _bbLWaveform, "bbLWaveform");
    registerChild ( _bbDetail, "bbDetail");
    registerChild ( _bbLConfigNew, "bbLConfigNew");

    //---- Start editable code block: DeckLTabbedDeck constructor

    //---- End editable code block: DeckLTabbedDeck constructor

}

DeckLTabbedDeck::~DeckLTabbedDeck()
{
    //---- Start editable code block: DeckLTabbedDeck destructor

    //---- End editable code block: DeckLTabbedDeck destructor

}

const char * DeckLTabbedDeck::className() // classname
{
    return ("DeckLTabbedDeck");
} // End className()

```


//---- Start editable code block: End of generated code

```
void DeckLTabbedDeck::set(ObjectManager *objMag)
{
    ((ObjectManager *)objMag) -> _deckL = this;
    ((ObjectManager *)objMag) -> set(_bbDisplay);
    ((ObjectManager *)objMag) -> set(_bbLROI);
    ((ObjectManager *)objMag) -> set(_bbHistogram);
    ((ObjectManager *)objMag) -> set(_bbDetail);
    ((ObjectManager *)objMag) -> set(_bbLPCMRA);
    ((ObjectManager *)objMag) -> set(_bbLWaveform);
    ((ObjectManager *)objMag) -> set(_bbLConfig);

    _bbDisplay -> set(objMag);
    _bbLROI -> set(objMag);
    _bbHistogram -> set(objMag);
    _bbLPCMRA -> set(objMag);
    _bbLWaveform -> set(objMag);
    _bbDetail -> set(objMag);
    _bbLConfig -> set(objMag);
    _bbLConfigNew -> set(objMag);
}
```

//---- End editable code block: End of generated code

```

////////////////////////////////////
//
// Source file for DeckRTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
// When you modify this source, limit your changes to
// modifying the sections between the
// "///---- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////

#include "DeckRTabbedDeck.h"
#include <Vk/VkDeck.h>

#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include "Bb3D.h"
#include "Bb3DLocalizer.h"
#include "BbAnimation.h"
#include "BbFlow.h"
#include "BbFlow3D.h"
#include "BbFormat.h"
#include "BbRHistogram.h"
#include "BbRROI.h"
#include "BbRTable.h"
#include "BbRWaveform.h"
#include "BbVelocity.h"
#include "BbVisual.h"
extern void VkUnimplemented(Widget, const char *);

//---- Start editable code block: headers and declarations
#include "ObjectManager.h"

//---- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String DeckRTabbedDeck::_defaultDeckRTabbedDeckResources[] = {

    //---- Start editable code block: DeckRTabbedDeck Default Resources

    //---- End editable code block: DeckRTabbedDeck Default Resources

    (char*)NULL
};

```

```
//---- DeckRTabbedDeck Constructor
```

```
DeckRTabbedDeck::DeckRTabbedDeck ( const char *name, Widget parent ) :
    VkTabbedDeck ( name, parent )
{
    // Load any class-default resources for this object

    setDefaultResources ( baseWidget(), _defaultDeckRTabbedDeckResources );

    _bbRHistogram = new BbRHistogram( "bbRHistogram", _vkdeck->baseWidget() );
    _bbRHistogram->show();

    _bbFlow = new BbFlow( "bbFlow", _vkdeck->baseWidget() );
    _bbFlow->show();

    _bbVisual = new BbVisual( "bbVisual", _vkdeck->baseWidget() );
    _bbVisual->show();

    _bbFormat = new BbFormat( "bbFormat", _vkdeck->baseWidget() );
    _bbFormat->show();

    _bbFlow3D = new BbFlow3D( "bbFlow3D", _vkdeck->baseWidget() );
    _bbFlow3D->show();

    _bb3DLocalizer = new Bb3DLocalizer( "bb3DLocalizer", _vkdeck->baseWidget() );
    _bb3DLocalizer->show();

    _bbVelocity = new BbVelocity( "bbVelocity", _vkdeck->baseWidget() );
    _bbVelocity->show();

    _bb3D = new Bb3D( "bb3D", _vkdeck->baseWidget() );
    _bb3D->show();

    _bbAnimation = new BbAnimation( "bbAnimation", _vkdeck->baseWidget() );
    _bbAnimation->show();

    _bbRROI = new BbRROI( "bbRROI", _vkdeck->baseWidget() );
    _bbRROI->show();

    _bbRWaveform = new BbRWaveform( "bbRWaveform", _vkdeck->baseWidget() );
    _bbRWaveform->show();

    _bbRTable = new BbRTable( "bbRTable", _vkdeck->baseWidget() );
    _bbRTable->show();

    registerChild ( _bbRHistogram, "bbRHistogram");
    registerChild ( _bbFlow, "bbFlow");
    registerChild ( _bbVisual, "bbVisual");
    registerChild ( _bbFormat, "bbFormat");
    registerChild ( _bbFlow3D, "bbFlow3D");
    registerChild ( _bb3DLocalizer, "bb3DLocalizer");
    registerChild ( _bbVelocity, "bbVelocity");
    registerChild ( _bb3D, "bb3D");
    registerChild ( _bbAnimation, "bbAnimation");
    registerChild ( _bbRROI, "bbRROI");
    registerChild ( _bbRWaveform, "bbRWaveform");
    registerChild ( _bbRTable, "bbRTable");

    //---- Start editable code block: DeckRTabbedDeck constructor

    //---- End editable code block: DeckRTabbedDeck constructor
```

```

}

DeckRTabbedDeck::~DeckRTabbedDeck()
{
    //---- Start editable code block: DeckRTabbedDeck destructor

    //---- End editable code block: DeckRTabbedDeck destructor
}

const char * DeckRTabbedDeck::className() // classname
{
    return ("DeckRTabbedDeck");
} // End className()

//---- Start editable code block: End of generated code

void DeckRTabbedDeck::set(ObjectManager *objMag)
{
    ((ObjectManager *)objMag) -> _deckR = this;

    ((ObjectManager *)objMag) -> set(_bbVisual);
    ((ObjectManager *)objMag) -> set(_bbRHistogram);
    ((ObjectManager *)objMag) -> set(_bbFlow);
    ((ObjectManager *)objMag) -> set(_bbRWaveform);
    ((ObjectManager *)objMag) -> set(_bbRTable);
    ((ObjectManager *)objMag) -> set(_bbAnimation);
    ((ObjectManager *)objMag) -> set(_bb3D);
    ((ObjectManager *)objMag) -> set(_bbVelocity);
    ((ObjectManager *)objMag) -> set(_bbRROI);
    (((ObjectManager *)objMag) -> set(_bbFormat);
    ((ObjectManager *)objMag) -> set(_bbFlow3D);
    ((ObjectManager *)objMag) -> set(_bb3DLocalizer);
    ((ObjectManager *)objMag) -> set(_bbFormat);

    _bbVisual -> set(objMag);
    _bbRHistogram -> set(objMag);
    _bbFlow -> set(objMag);
    _bbAnimation -> set(objMag);
    _bbRWaveform -> set(objMag);
    _bbRTable -> set(objMag);
    _bb3D -> set(objMag);
    _bbVelocity -> set(objMag);
    _bbRROI -> set(objMag);
    _bbFormat -> set(objMag);
    _bbFlow3D -> set(objMag);
    _bb3DLocalizer -> set(objMag);
}

//---- End editable code block: End of generated code

```

User: meide
Host: phoenix
Class: phoenix
Job: BbVelocityUI.C

571

```

////////////////////////////////////
//
// Source file for VkwindowMainWindow
//
// This class is a subclass of VkWindow
//
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Try to restrict any changes to the bodies of functions
// corresponding to menu items, the constructor and destructor.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
// Avoid gratuitous reformatting and other changes that might
// make it difficult to integrate changes made using RapidApp
////////////////////////////////////
#include "VkwindowMainWindow.h"

#include <Vk/VkApp.h>
#include <Vk/VkFileDialog.h>
#include <Vk/VkSubMenu.h>
#include <Vk/VkRadioSubMenu.h>
#include <Vk/VkMenuItem.h>
#include <Vk/VkMenuBar.h>
#include <Vk/VkResource.h>

// Externally defined classes referenced by this class:

#include "Bb.h"

extern void VkUnimplemented ( Widget, const char * );

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

// These are default resources for widgets in objects of this class
// All resources will be prepended by *<name> at instantiation,
// where <name> is the name of the specific instance, as well as the
// name of the baseWidget. These are only defaults, and may be overridden
// in a resource file by providing a more specific resource name

String VkwindowMainWindow::_defaultVkwindowMainWindowResources[] = {
    "closeButton.accelerator: Ctrl<Key>W",
    "closeButton.acceleratorText: Ctrl+W",
    "closeButton.labelString: Close",
    "closeButton.mnemonic: C",
    "copyButton.accelerator: Ctrl<Key>C",
    "copyButton.acceleratorText: Ctrl+C",
    "copyButton.labelString: Copy",
    "copyButton.mnemonic: C",
    "cutButton.accelerator: Ctrl<Key>X",
    "cutButton.acceleratorText: Ctrl+X",
    "cutButton.labelString: Cut",
    "cutButton.mnemonic: t",

```

```

"*editPane.labelString: Edit",
"*editPane.mnemonic: E",
"*exitButton.accelerator: Ctrl<Key>Q",
"*exitButton.acceleratorText: Ctrl+Q",
"*exitButton.labelString: Exit",
"*exitButton.mnemonic: x",
"*expert.labelString: Expert",
"*filePane.labelString: File",
"*filePane.mnemonic: F",
"*helpPane.labelString: Help",
"*helpPane.mnemonic: H",
"*help_click_for_help.accelerator: Shift<Key>F1",
"*help_click_for_help.acceleratorText: Shift+F1",
"*help_click_for_help.labelString: Click For Help",
"*help_click_for_help.mnemonic: C",
"*help_index.labelString: Index",
"*help_index.mnemonic: I",
"*help_keys_and_short.labelString: Keys and Shortcuts",
"*help_keys_and_short.mnemonic: K",
"*help_overview.labelString: Overview",
"*help_overview.mnemonic: O",
"*help_prod_info.labelString: Product Information",
"*help_prod_info.mnemonic: P",
"*imgInfo.labelString: Image Info",
"*mraInfo.labelString: MR Angio Info",
"*newButton.accelerator: Ctrl<Key>N",
"*newButton.acceleratorText: Ctrl+N",
"*newButton.labelString: New",
"*newButton.mnemonic: N",
"*novies.labelString: Novies",
"*openButton.accelerator: Ctrl<Key>O",
"*openButton.acceleratorText: Ctrl+O",
"*openButton.labelString: Open Flow",
"*openButton.mnemonic: O",
"*pasteButton.accelerator: Ctrl<Key>V",
"*pasteButton.acceleratorText: Ctrl+V",
"*pasteButton.labelString: Paste",
"*pasteButton.mnemonic: P",
"*pcmraLocalizer.labelString: PCMRA Cut",
"*printButton.accelerator: Ctrl<Key>P",
"*printButton.acceleratorText: Ctrl+P",
"*printButton.labelString: Print",
"*printButton.mnemonic: P",
"*saveButton.accelerator: Ctrl<Key>S",
"*saveButton.acceleratorText: Ctrl+S",
"*saveButton.labelString: Save",
"*saveButton.mnemonic: S",
"*saveasButton.labelString: Save As Flow",
"*saveasButton.mnemonic: A",
"*theVkUndoManagerButton.accelerator: Ctrl<Key>Z",
"*theVkUndoManagerButton.acceleratorText: Ctrl+Z",
"*theVkUndoManagerButton.labelString: Undo",
"*theVkUndoManagerButton.mnemonic: U",
"*user.labelString: User",
"*user.mnemonic: V",
"*viewPane.labelString: View",
"*viewPane.mnemonic: V",

//---- Start editable code block: VkwindowMainWindow Default Resources

//---- End editable code block: VkwindowMainWindow Default Resources

(char*)NULL
};

```

```
VkwindowMainWindow::VkwindowMainWindow ( const char *name,
                                         ArgList args,
                                         Cardinal argCount) :
    VkWindow ( name, args, argCount )
{
    // Load any class-default resources for this object

    setDefaultResources ( baseWidget(), _defaultVkwindowMainWindowResources );

    // Create the view component contained by this window

    _bb = new Bb ( "bb",mainWindowWidget() );

    XtVaSetValues ( _bb->baseWidget(),
                    XmNwidth, 1250,
                    XmNheight, 949,
                    (XtPointer) NULL );

    // Add the component as the main view

    addView ( _bb );

    // Create the panes of this window's menubar. The menubar itself
    // is created automatically by ViewKit

    // The filePane menu pane

    _filePane = addMenuPane ( "filePane" );

    _newButton = _filePane->addAction ( "newButton",
                                       &VkwindowMainWindow::newFileCallback,
                                       (XtPointer) this );

    _openButton = _filePane->addAction ( "openButton",
                                       &VkwindowMainWindow::openFileCallback,
                                       (XtPointer) this );

    _saveButton = _filePane->addAction ( "saveButton",
                                       &VkwindowMainWindow::saveCallback,
                                       (XtPointer) this );

    _saveasButton = _filePane->addAction ( "saveasButton",
                                       &VkwindowMainWindow::saveasCallback,
                                       (XtPointer) this );

    _printButton = _filePane->addAction ( "printButton",
                                       &VkwindowMainWindow::printCallback,
                                       (XtPointer) this );

    _separator1 = _filePane->addSeparator();

    _closeButton = _filePane->addAction ( "closeButton",
                                       &VkwindowMainWindow::closeCallback,
                                       (XtPointer) this );

    _exitButton = _filePane->addAction ( "exitButton",
                                       &VkwindowMainWindow::quitCallback,
                                       (XtPointer) this );

    // The editPane menu pane
```



```

_editPane = addMenuPane ( "editPane" );

_editPane->add ( theUndoManager );

_cutButton = _editPane->addAction ( "cutButton",
                                   &VkwindowMainWindow::cutCallback,
                                   (XtPointer) this );

_copyButton = _editPane->addAction ( "copyButton",
                                   &VkwindowMainWindow::copyCallback,
                                   (XtPointer) this );

_pasteButton = _editPane->addAction ( "pasteButton",
                                   &VkwindowMainWindow::pasteCallback,
                                   (XtPointer) this );

// The viewPane menu pane
_viewPane = addMenuPane ( "viewPane" );

_imgInfo = _viewPane->addAction ( "imgInfo",
                                   &VkwindowMainWindow::imgInfoCallbackCallback,
                                   (XtPointer) this );

_mraInfo = _viewPane->addAction ( "mraInfo",
                                   &VkwindowMainWindow::mraInfoCallbackCallback,
                                   (XtPointer) this );

_pcmraLocalizer = _viewPane->addAction ( "pcmraLocalizer",
                                   &VkwindowMainWindow::pcmraCutCallbackCallback,
                                   (XtPointer) this );

// The user menu pane
_user = addMenuPane ( "user" );

_novies = _user->addAction ( "novies",
                             &VkwindowMainWindow::noviesCallbackCallback,
                             (XtPointer) this );

_expert = _user->addAction ( "expert",
                             &VkwindowMainWindow::expertCallbackCallback,
                             (XtPointer) this );

//---- Start editable code block: VkwindowMainWindow constructor

//---- End editable code block: VkwindowMainWindow constructor

} // End Constructor

VkwindowMainWindow::~VkwindowMainWindow()
{
    delete _bb;
    //---- Start editable code block: VkwindowMainWindow destructor

    //---- End editable code block: VkwindowMainWindow destructor
} // End destructor

const char *VkwindowMainWindow::className()

```

[illegible]

[illegible]

```

VkwindowMainWindow* obj = ( VkwindowMainWindow * ) clientData;
obj->quit ( w, callData );
}

void VkwindowMainWindow::saveCallback ( Widget w,
                                       XtPointer clientData,
                                       XtPointer callData )
{
    VkwindowMainWindow* obj = ( VkwindowMainWindow * ) clientData;
    obj->save ( w, callData );
}

void VkwindowMainWindow::saveasCallback ( Widget w,
                                          XtPointer clientData,
                                          XtPointer callData )
{
    VkwindowMainWindow* obj = ( VkwindowMainWindow * ) clientData;
    obj->saveas ( w, callData );
}

////////////////////////////////////
// The following functions are called from callbacks
////////////////////////////////////

void VkwindowMainWindow::close ( Widget, XtPointer )
{
    //---- Start editable code block: close

    // To close a window, just delete the object
    // checking first with the view object.
    // If this is the last window, ViewKit will exit

    if(okToQuit())
        delete this;

    //---- End editable code block: close
} // End VkwindowMainWindow::close()

void VkwindowMainWindow::copy ( Widget w, XtPointer callData )
{
    //---- Start editable code block: VkwindowMainWindow copy

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    // By default this member function passes control
    // to the component contained by this window

    _bb->copy();

    //---- End editable code block: VkwindowMainWindow copy
} // End VkwindowMainWindow::copy()

void VkwindowMainWindow::cut ( Widget w, XtPointer callData )
{
    //---- Start editable code block: VkwindowMainWindow cut

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

```

```
// By default this member function passes control
// to the component contained by this window
```

579

```
_bb->cut();
```

```
//---- End editable code block: VkwindowMainWindow cut
```

```
} // End VkwindowMainWindow::cut()
```

```
void VkwindowMainWindow::expertCallback ( Widget w, XtPointer callData )
{
```

```
//---- Start editable code block: VkwindowMainWindow expertCallback
```

```
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
// By default this member function passes control
// to the component contained by this window
```

```
_bb->expertCallback( w, callData);
```

```
//---- End editable code block: VkwindowMainWindow expertCallback
```

```
} // End VkwindowMainWindow::expertCallback()
```

```
void VkwindowMainWindow::imgInfoCallback ( Widget w, XtPointer callData )
{
```

```
//---- Start editable code block: VkwindowMainWindow imgInfoCallback
```

```
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
// By default this member function passes control
// to the component contained by this window
```

```
_bb->imgInfoCallback( w, callData);
```

```
//---- End editable code block: VkwindowMainWindow imgInfoCallback
```

```
} // End VkwindowMainWindow::imgInfoCallback()
```

```
void VkwindowMainWindow::mraInfoCallback ( Widget w, XtPointer callData )
{
```

```
//---- Start editable code block: VkwindowMainWindow mraInfoCallback
```

```
XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;
```

```
// By default this member function passes control
// to the component contained by this window
```

```
_bb->mraInfoCallback( w, callData);
```

```
//---- End editable code block: VkwindowMainWindow mraInfoCallback
```

```
} // End VkwindowMainWindow::mraInfoCallback()
```

```
void VkwindowMainWindow::newFile ( Widget w, XtPointer callData )
```

```
{
//---- Start editable code block: VkwindowMainWindow newFile
```

```

XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

// By default this member function passes control
// to the component contained by this window

_bb->newFile();

//---- End editable code block: VkwindowMainWindow newFile
} // End VkwindowMainWindow::newFile()

void VkwindowMainWindow::noviesCallback ( Widget w, XtPointer callData )
{
    //---- Start editable code block: VkwindowMainWindow noviesCallback

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    // By default this member function passes control
    // to the component contained by this window

    _bb->noviesCallback( w, callData);

    //---- End editable code block: VkwindowMainWindow noviesCallback
} // End VkwindowMainWindow::noviesCallback()

void VkwindowMainWindow::openFile ( Widget, XtPointer )
{
    //---- Start editable code block: VkwindowMainWindow openFile

    // This virtual function is called from openFileCallback
    // Use the blocking mode of the file selection dialog
    // to get a file

    theFileSelectionDialog -> setDirectory("/usr/people/canvas/active_patients");
    if(theFileSelectionDialog->postAndWait() == VkDialogManager::OK)
    {
        _bb->openFile(theFileSelectionDialog->fileName());
    }

    //---- End editable code block: VkwindowMainWindow openFile
} // End VkwindowMainWindow::openFile()

void VkwindowMainWindow::paste ( Widget w, XtPointer callData )
{
    //---- Start editable code block: VkwindowMainWindow paste

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    // By default this member function passes control
    // to the component contained by this window

    _bb->paste();

    //---- End editable code block: VkwindowMainWindow paste
} // End VkwindowMainWindow::paste()

void VkwindowMainWindow::pcmraCutCallback ( Widget w, XtPointer callData )

```

```

{
    //---- Start editable code block: VkwindowMainWindow pcmraCutCallback    581
    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    // By default this member function passes control
    // to the component contained by this window

    _bb->pcmraCutCallback( w, callData);

    //---- End editable code block: VkwindowMainWindow pcmraCutCallback
}    // End VkwindowMainWindow::pcmraCutCallback()

void VkwindowMainWindow::print ( Widget, XtPointer )
{
    //---- Start editable code block:  print

    // This virtual function is called from printCallback

    _bb->print(NULL);

    //---- End editable code block:  print
}    // End VkwindowMainWindow::print()

void VkwindowMainWindow::quit ( Widget, XtPointer )
{
    // Exit via quitYourself() to allow the application
    // to go through its normal shutdown routines, checking with
    // all windows, views, and so on.

    theApplication->quitYourself();
}    // End VkwindowMainWindow::quit()

void VkwindowMainWindow::save ( Widget w, XtPointer callData )
{
    //---- Start editable code block: VkwindowMainWindow save

    XmPushButtonCallbackStruct *cbs = (XmPushButtonCallbackStruct*) callData;

    // By default this member function passes control
    // to the component contained by this window

    _bb->save();

    //---- End editable code block: VkwindowMainWindow save
}    // End VkwindowMainWindow::save()

void VkwindowMainWindow::saveas ( Widget, XtPointer )
{
    //---- Start editable code block: VkwindowMainWindow saveas

    // This virtual function is called from saveasCallback
    // Use the blocking mode of the file selection dialog
    // to get a file

    if(theFileSelectionDialog->postAndWait() == VkDialogManager::OK)

```

```
{
    _bb->saveas(theFileSelectionDialog->fileName());
}

//----- End editable code block: Vmainwindow saveas
} // End Vmainwindow::saveAs()


//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code
```



```
////////////////////////////////////  
// This file contains a single global function "VkUnimplemented".  
// This function supports development using Fix+Continue.  
// You can simply set a breakpoint in this function to stop in  
// all unimplemented callback functions in a program.  
////////////////////////////////////
```

```
#include <iostream.h>
```

```
#include <Xm/Xm.h>
```

```
void VkUnimplemented(Widget w, const char *str)
```

```
{  
    cerr << "The member function " << str << "()  was invoked";  
    if ( w )  
        cerr << " by " << XtName(w);  
    cerr << endl << flush;  
}
```

User: meide
Host: phoenix
Class: phoenix
Job: VkwindowMainWindow.C

584

```

////////////////////////////////////
-//
// Header file for DrawingAreaUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, DrawingArea is instantiated
//
-// To extend or alter the behavior of this class, you should
// modify the DrawingArea files
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
-// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef DRAWINGAREUI_H
#define DRAWINGAREUI_H
#include <Vk/VkComponent.h>

-///---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class DrawingAreaUI : public VkComponent
{
    public:

        DrawingAreaUI ( const char *, Widget , int);
        DrawingAreaUI ( const char * );
        ~DrawingAreaUI();
        void create ( Widget, int );
        const char * className();

        ///---- Start editable code block: DrawingArea public

        Widget baseWidget2() {return _baseWidget2;}
        Widget _baseWidget2;

        ///---- End editable code block: DrawingArea public

    protected:

        // Widgets created by this class

        Widget _drawingArea;

        // These virtual functions are called from the private callbacks (below)
        // Intended to be overridden in derived classes to define actions

```

```

virtual void expose ( Widget, XtPointer );
virtual void input ( Widget, XtPointer );
virtual void resize ( Widget, XtPointer );

//---- Start editable code block: DrawingArea protected

virtual void motion ( Widget, XEvent * );

//---- End editable code block: DrawingArea protected

private:

// Array of default resources

static String      _defaultDrawingAreaUIResources[];

// Callbacks to interface with Motif

static void exposeCallback ( Widget, XtPointer, XtPointer );
static void inputCallback ( Widget, XtPointer, XtPointer );
static void resizeCallback ( Widget, XtPointer, XtPointer );

//---- Start editable code block: DrawingArea private

static void motion(Widget w, XtPointer clientData,
                  XEvent *event, Boolean *flag);

//---- End editable code block: DrawingArea private
};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```

```

////////////////////////////////////
//
// Header file for DrawingArea
//
// This file is generated by RapidApp 1.2
//
// This class is derived from DrawingAreaUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef DRAWINGAREA_H
#define DRAWINGAREA_H
#include "DrawingAreaUI.h"
//---- Start editable code block: headers and declarations
#include "Utility_Vision.h"
//---- End editable code block: headers and declarations

//---- DrawingArea class declaration

class DrawingArea : public DrawingAreaUI
{
public:

    DrawingArea ( const char * );
    DrawingArea ( const char *, Widget, int );
    DrawingArea(int w, int h, unsigned char **grayimg,
        const char *name, Widget parent, int flag) ;
    DrawingArea(int w, int h, const char *name, Widget parent, int flag);

    ~DrawingArea();
    const char * className();

    static VkComponent *CreateDrawingArea( const char *name, Widget parent, int );

//---- Start editable code block: DrawingArea public

    int get_width() {return width;}
    int get_height() {return height;}

    int get_depth() {return _depth;}
    Pixel get_offset() {return _offset;}

    void set_depth(int depth) {_depth = depth;}
    void set_offset(Pixel offset) {_offset = offset;}

    void set_imgdata(unsigned char **grayimg);
    void set_imgdata(int w,int h, unsigned char **grayimg);

    void set_imgdata(ColorImage *img);
    void set_imgdata(int w,int h, ColorImage *img);

```

```
void display();
void display(int x, int y);
void display(int x, int y, int w, int h);
void set_Origin(int x, int y);
```

```
void copyArea(int x, int y, int w, int h);
```

```
//---- End editable code block: DrawingArea public
```

protected:

```
// These functions will be called as a result of callbacks
// registered in DrawingAreaUI
```

```
virtual void expose ( Widget, XtPointer );
virtual void input ( Widget, XtPointer );
virtual void resize ( Widget, XtPointer );
```

```
//---- Start editable code block: DrawingArea protected
```

```
virtual void motion ( Widget, XEvent * );
```

```
void MakeColormap(Widget w);
void clear_memory();
```

```
void create_pixmap(int w, int h, unsigned char **grayimg);
void create_pixmap(int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
Pixmap get_pixmap2(int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
```

```
void setXData(int bw, int depth, Pixel offset,
    int r, int g, int b, int pos, unsigned char *img);
```

```
unsigned char *toXdata(int bw, int w, int h, unsigned char **grayimg);
unsigned char *toXdata(int bw, int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
```

```
XImage *img2XImage(Display *theDisplay, Screen *theScreen,
    unsigned char *image_data, int image_width,
    int image_height, int depth);
```

```
int      _depth;
Pixel    _offset;
Pixmap   _pixmap;
GC       _gc;
XImage   *_ximage;
```

```
//---- End editable code block: DrawingArea protected
```

private:

```
static void* RegisterDrawingAreaInterface();
```

```
//---- Start editable code block: DrawingArea private
```

```
int      width,height;
```

```
//---- End editable code block: DrawingArea private
```

```
};  
//---- Start editable code block: End of generated code  
..  
//---- End editable code block: End of generated code  
#endif
```

```

////////////////////////////////////
//
// Header file for DrawingArea
//
// This file is generated by RapidApp 1.2
//
// This class is derived from DrawingAreaUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef DRAWINGAREA_H
#define DRAWINGAREA_H
#include "DrawingAreaUI.h"
//---- Start editable code block: headers and declarations
#include "Utility_Vision.h"
//---- End editable code block: headers and declarations

//---- DrawingArea class declaration

class DrawingArea : public DrawingAreaUI
{
public:
    DrawingArea ( const char * );
    DrawingArea ( const char *, Widget, int );
    DrawingArea(int w, int h, unsigned char **grayimg,
        const char *name, Widget parent, int flag) ;
    DrawingArea(int w, int h, const char *name, Widget parent, int flag);

    ~DrawingArea();
    const char * className();

    static VkComponent *CreateDrawingArea( const char *name, Widget parent, int );

//---- Start editable code block: DrawingArea public

    int get_width() {return width;}
    int get_height() {return height;}

    int get_depth() {return _depth;}
    Pixel get_offset() {return _offset;}

    void set_depth(int depth) {_depth = depth;}
    void set_offset(Pixel offset) {_offset = offset;}

    void set_imgdata(unsigned char **grayimg);
    void set_imgdata(int w,int h, unsigned char **grayimg);

    void set_imgdata(ColorImage *img);
    void set_imgdata(int w,int h, ColorImage *img);

```



```
void display();
void display(int x, int y);
void display(int x, int y, int w, int h);
void set_Origin(int x, int y);

void copyArea(int x, int y, int w, int h);
```

```
//---- End editable code block: DrawingArea public
```

protected:

```
// These functions will be called as a result of callbacks
// registered in DrawingAreaUI
```

```
virtual void expose ( Widget, XtPointer );
virtual void input ( Widget, XtPointer );
virtual void resize ( Widget, XtPointer );
```

```
//---- Start editable code block: DrawingArea protected
```

```
virtual void motion ( Widget, XEvent * );
```

```
void MakeColormap(Widget w);
void clear_memory();
```

```
void create_pixmap(int w, int h, unsigned char **grayimg);
void create_pixmap(int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
Pixmap get_pixmap2(int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
```

```
void setXData(int bw, int depth, Pixel offset,
    int r, int g, int b, int pos, unsigned char *img);
```

```
unsigned char *toXdata(int bw, int w, int h, unsigned char **grayimg);
unsigned char *toXdata(int bw, int w, int h, unsigned char **r,
    unsigned char **g, unsigned char **b);
```

```
XImage *img2XImage(Display *theDisplay, Screen *theScreen,
    unsigned char *image_data, int image_width,
    int image_height, int depth);
```

```
int      _depth;
Pixel    _offset;
Pixmap   _pixmap;
GC       _gc;
XImage   *_ximage;
```

```
//---- End editable code block: DrawingArea protected
```

private:

```
static void* RegisterDrawingAreaInterface();
```

```
//---- Start editable code block: DrawingArea private
```

```
int      width,height;
```

```
//---- End editable code block: DrawingArea private
```

```
};  
//---- Start editable code block: End of generated code  
  
//---- End editable code block: End of generated code  
#endif
```

```

////////////////////////////////////
// MedDrawingArea.h
////////////////////////////////////
#ifndef MEDDRAWINGAREA_H
#define MEDDRAWINGAREA_H

#include "DrawingArea.h"
#include "Utility_Vision.h"

class MedDrawingArea : public DrawingArea {

public:

    MedDrawingArea(const char *name, Widget parent, int flag) ;
    ~MedDrawingArea();

    void set(int w, int h, short **img, int visual_method,
             int scale_method, float zoom, float winCenter, float winWidth, int flowDir=0);
    void setData(int w, int h, short **img, int visual_method,
                int scale_method, float zoom, float winCenter, float winWidth, int flowDir=0);

    Boolean update_visual(int visual_method);
    Boolean update(float center, float width);
    Boolean update(int scale_method);
    Boolean update(float zoom);

    Pixmap  get_pixmap(short **img);

//
// Data Members
//
    Boolean  _button1Pressed;

    short   **_orgImg;
    int      _orgWidth, _orgHeight;

    int      _visual_method;  // Gray or Color
    int      _flowDir;        // Flow Direction

    int      _scale_method;   // Resampling Method

    float     _zoom;  // the original size 16-bit image ImgGE::imgdata is
                     // resampled by zoom factor.
    short     **_zoomImg; // 16-bit zoomed image
                     // 8-bit unsigned char **grayimg for viewing
                     // is obtained from
                     // the 16-bit zoomed image by applying
                     // Width and Level, i.e., Window Width and Center.

    float **getFloatImg();

    float     _winCenter;
    float     _winWidth;

    ColorImage *_cimg;
    ColorImage *toVisual(int visual_method, int w, int h, short **shimg, float pl, fl

    void semiFlow(int minI, int maxI, unsigned char **mask);
    void semiFlow2(int minI, int maxI, unsigned char **mask);
    ColorImage *_cimg2;
    float _minFlow, _maxFlow;
    void get_mmFlow(unsigned char **mask, float *minI, float *maxI);

    void remove_cimg();
    void create_cimg2();

```

```
void highlight(float percent, unsigned char r1, unsigned char g1, unsigned char b1
    unsigned char *r, unsigned char *g2, unsigned char *b2 );
```

594

protected:

```
virtual void expose(Widget, XtPointer);
virtual void input(Widget, XtPointer);
virtual void motion ( Widget, XEvent * );
```

};

#endif

```

////////////////////////////////////
// ROIMedDrawingArea.h
////////////////////////////////////
#ifndef ROIMEDDRAWINGAREA_H
#define ROIMEDDRAWINGAREA_H

#include "MedDrawingArea.h"
#include "ROI.h"

#include "ObjectManager.h"

class ROIMedDrawingArea : public MedDrawingArea {

public:

    ROIMedDrawingArea(const char *name, Widget parent, int);
    ~ROIMedDrawingArea();

    int      _roi_action;
    int      _roi_type;
    int      _roi_color;
    ROI      *_ROI;

    void      pressed(int xpos, int ypos);
    void      released(int xpos, int ypos);
    void      moved(int xpos, int ypos);
    void      finished(int xpos, int ypos);

    void      midpressed(int xpos, int ypos);
    void      midmoved(int xpos, int ypos);

    void      CreateROI(int roi_type);
    void      CreateROI2(int roi_type);
    void      AcceptROI();
    void      EraseROI();
    void      ShowROI();
    void      HideROI();

    unsigned char **get_mask();
    void show_info(int x, int y);

    ObjectManager *_objMag;
    void setObj(ObjectManager *objMag) {_objMag = objMag;}

    void display();

    void update(float center, float width) { if(MedDrawingArea::update(center, width))
    void update(int scale_method) { if(MedDrawingArea::update(scale_method)) ShowROI()
    void update(float zoom) { if(MedDrawingArea::update(zoom)) ShowROI(); }
    void update_visual(int visual_method) { if(MedDrawingArea::update_visual(visual_me

    void copyArea(int x, int y, int w, int h);

    void set_color(int);

    XImage *get_XImage();

protected:

    Boolean _button2Pressed;

    virtual void expose(Widget, XtPointer);
    virtual void input(Widget, XtPointer);
    virtual void motion ( Widget, XEvent * );

```

};

#endif

User: meide
Host: phoenix
Class: phoenix
Job: DrawingAreaUI.h

597

```

////////////////////////////////////
// LineDrawingArea.h
////////////////////////////////////
#ifndef LINEDRAWINGAREA
#define LINEDRAWINGAREA

#include "DrawingAreaUI.h"
#include "Utility_Widget.h"

#define DRAW_BAR 0
#define DRAW_CURVE 1

class LineDrawingArea : public DrawingAreaUI {
public:
    LineDrawingArea(int w, int h, const char *name, Widget parent, int type=DRAW_BAR,
~LineDrawingArea());

    void set(int sz, float *x, float *y);

    void display(int x, int y);
    void display(int color=COLOR_GREEN);

    void draw_bar(int color);
    void draw_curve(int color);

    void draw_onePoint(int i, GC gc);

//
// Data Members
//
    Boolean _button1Pressed;
    int _draw_type;
    int _width, _height; // The size of DrawingArea

    int _size; // The size of Array x and y
    float *_x; // Array x --- x[0],x[1],...,x[_size-1]
    float *_y; // Array y --- y[0],y[1],...,y[_size-1]

    float _minX, _maxX;
    float _minY, _maxY;

    int *_drawX;
    int *_drawY;

protected:
    virtual void expose(Widget, XtPointer);
    virtual void resize(Widget, XtPointer);
    virtual void input(Widget, XtPointer);

    virtual void motion ( Widget w, XEvent *event );

private:
};

#endif

```



```

////////////////////////////////////
// TwoLinesLineDrawingArea
////////////////////////////////////
#ifndef TWOLINESLINEDRAWINGAREA_H
#define TWOLINESLINEDRAWINGAREA_H

#include "LineDrawingArea.h"
#include "TwoLines.h"

class TwoLinesLineDrawingArea : public LineDrawingArea {

public:

    TwoLinesLineDrawingArea(int w, int h, const char *name, Widget parent, int type=DF
    ~TwoLinesLineDrawingArea();

    void newTwoLines(float center, float width, float minI, float maxI);
    void set(int x1, int x2) { _twolines->set(int(x1), int(x2)); }

//
// Data Members
//
    TwoLines *_twolines;

protected:

    virtual void expose(Widget, XtPointer);
    virtual void resize(Widget, XtPointer);
    virtual void input(Widget, XtPointer);

    virtual void motion ( Widget w, XEvent *event );

private:

};

#endif

```

```

////////////////////////////////////
// HistoTwoLinesDrawingArea.h
////////////////////////////////////
#ifndef HISTOTWOLINESDRAWINGAREA_H
#define HISTOTWOLINESDRAWINGAREA_H

#include "TwoLinesLineDrawingArea.h"

#include "ObjectManager.h"
#include "MedDrawingArea.h"

class HistoTwoLinesDrawingArea : public TwoLinesLineDrawingArea {
public:
    HistoTwoLinesDrawingArea(int w, int h, const char *name, Widget parent, int type=I
    ~HistoTwoLinesDrawingArea();

    void newTwoLines(float center, float width);
    void set(int w, int h, short **img, int size=512, unsigned char **mask=NULL, float

    void change();
    void update_lowhigh(float center, float width);
    void update_map();
    void update_imgView();

//
// Data Members
//
    int    _w, _h;
    short  **_img;
    float  _minI, _maxI;

    short  **_mapImg;
    MedDrawingArea *_map;

    int    _whoami;
    ObjectManager *_objMag;
    void set(ObjectManager *objMag, int whoami) {_objMag = objMag; _whoami = whoami;}

    Widget  _label_min;
    Widget  _label_max;
    Widget  _label_low;
    Widget  _label_high;
    void set(Widget w1, Widget w2, Widget w3, Widget w4)
        {_label_min = w1, _label_max = w2, _label_low = w3, _label_high = w4;}

protected:

    virtual void expose(Widget, XtPointer);
    virtual void resize(Widget, XtPointer);
    virtual void input(Widget, XtPointer);
    virtual void motion ( Widget w, XEvent *event );

    void set_mm();
    void set_lowhigh();

    float  *get_histogram(int w, int h, short **img, int size,
        float low_img, float high_img, float *minI, float *maxI, unsigned char **mas

// input :      w, h, img[h][w]
//            low_img,high_img
//
// Calculate
// [low,high] = [min,max] AND [low_img,high_img]

```

```
//      where:
//          min = { img }
//          max = max { img }
//      Formula:
//          if(min < low_img) low = low_img
//          else low = min
//
//          if(max > high_img) high = high_img
//          else high = max
//
//
//      output:      size,histo[size],
//                  *histo_max = max( histo[i], i=0,...,size-1 )
//
//                  [0,size-1]  <-->  [low,high]
//
private:

};

#endif
```

```
#ifndef UTILITY_MATH_H
#define UTILITY_MATH_H
```

602

```
#include <math.h>
```

```
class Utility_Math
{
```

```
public:
```

```
    Utility_Math ();
```

```
    ~Utility_Math();
```

```
    int int_t(float x);
```

```
    float distance(float x1, float y1, float x2, float y2) { return sqrtf((x2 - x1) * (
```

```
    void get_minmax(int sz, float *x, float *minX, float *maxX);
```

```
    int solve_poly2(float a, float b, float c, float *x1, float *x2);
```

```
    //
```

```
    // Given: Polynomial Equation:  $a x^2 + b x + c = 0$ 
```

```
    // a, b, and c
```

```
    //
```

```
    // Find: x (x1 and x2)
```

```
    //
```

```
    int lineParaFromTwoPoints(float x1, float y1, float x2, float y2,
    float *c1, float *c2);
```

```
    //
```

```
    // Given: Linear Equation:  $y = c1 * x + c2$ 
```

```
    // Two Points: (y1,x1) and (y2,x2)
```

```
    //
```

```
    // Find: c1 and c2
```

```
    //
```

```
    // if (x2 == x1) then Linear Equation:  $x = *c1$  and return 0
```

```
    // o.w. return 1
```

```
    //
```

```
    int lineParaFromTwoPoints(float x1, float y1, float sita,
    float *c1, float *c2);
```

```
    //
```

```
    // Given: Linear Equation:  $y = c1 * x + c2$ 
```

```
    // one Points: (y1,x1)
```

```
    // angle: sita
```

```
    //
```

```
    // Find: c1 and c2
```

```
    //
```

```
    // if (sita == pi/2) then Linear Equation:  $x = *c1$  and return 0
```

```
    // o.w. return 1
```

```
    //
```

```
    //
```

```
    float get_angle(float x1, float y1, float x2, float y2);
```

```
    //
```

```
    // find the angle of the line connecting the given two points
```

```
    //
```

```
private:
```

```
};
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: LineDrawingArea.h

603

```

#ifndef UTILITY_VISION_H
#define UTILITY_VISION_H

#include <stdio.h>

#define SCALE_SPLINE 1
#define SCALE_SIMPLE 2

#define VISUAL_GRAY 1
#define VISUAL_COLOR 2

typedef struct {
    unsigned char **red;
    unsigned char **green;
    unsigned char **blue;
} ColorImage;

class Utility_Vision
{
public:
    Utility_Vision ();
    ~Utility_Vision();

    void get_bound(int w, int h, short **img, float *min_I, float *max_I);

    ColorImage *toVisual(int visual_method, int w, int h, short **shimg, float pl, float pr);
    unsigned char **toGray(int w, int h, short **shimg, float widCenter, float widWidth);
    ColorImage *toColor(int w, int h, short **shimg, float low, float high) ;
    ColorImage *toColor2(int w, int h, short **shimg, float low, float high) ;
    short **scale_img(int, int w1, int h1, short **img1, float zoom, int *w2, int *h2);
    short **copy_img(int w, int h, short **img);
    short **shrinking_img(int w1, int h1, short **in_img, int w2, int h2);
    short **stretching_img(int w1, int h1, short **in_img, int w2, int h2);
    inline short bilinear(float dx, float dy, int x1, int y1, short **img);
    short **simple_stretching(int w1, int h1, short **img1, int *w2, int *h2);

    void get_ROI(short **img, int x, int y, int w, int h, short **imgdata);
    short **get_ROI(short **img, int x, int y, int w, int h, unsigned char **mask = NULL);

    void highlight(float percent, int r1, int g1, int b1,
        int *r2, int *g2, int *b2 );

    void freeImg(unsigned char **grayimg);
    void freeCImg(ColorImage *img);
    void freeShimg(short **img);

private:
};

#endif

```

```
#ifndef UTILITY_WIDGET_H
#define UTILITY_WIDGET_H
```

604

```
#include <Vk/VkComponent.h>
#include "Utility_Vision.h"
```

```
#define COLOR_RED 0
#define COLOR_GREEN 1
#define COLOR_BLUE 2
#define COLOR_BLACK 3
#define COLOR_WHITE 4
#define COLOR_YELLOW 5
```

```
class Utility_Widget
{
```

```
public:
```

```
Utility_Widget ();
~Utility_Widget();
```

```
GC get_xorGC(Widget w);
GC get_GC(Widget w, int mode=COLOR_RED);
GC get_GC(Widget w, unsigned char r, unsigned char g=0, unsigned char b=0);
```

```
unsigned char **get_mask(Widget wid, int w, int h);
```

```
void set_label(Widget label, int);
void set_label(Widget label, float);
void set_label(Widget label, char *);
```

```
void set_textfield(Widget textfield, int i);
void set_textfield(Widget textfield, float);
```

```
void draw_point(Widget w, GC gc, float x, float y);
void draw_line(Widget w, GC gc, float x1, float y1, float x2, float y2);
void draw_rectangle(Widget wid, GC gc, float x, float y, float w, float h);

void draw_point(Widget w, float x, float y, int w1, int h1, ColorImage *cimg);
```

```
private:
```

```
};
```

```
#endif
```

```
#ifndef UTILITY_H
#define UTILITY_H
```

605

```
#include <Vk/VkComponent.h>
#include "GE.h"
#include "ImgAlloc.h"
#include "Point.h"
#include "Flow.h"
```

```
#define MY_LEFT 0
#define MY_RIGHT 1
```

```
#define ROI_LEFT 0
#define ROI_RIGHT 1
```

```
#define LEFT_MAX_WIDTH 512
#define LEFT_MAX_HEIGHT 512
#define RIGHT_MAX_WIDTH 512
#define RIGHT_MAX_HEIGHT 512
```

```
#define IMAGE_NONE 0
#define IMAGE_CT 1
#define IMAGE_MR 2
#define IMAGE_PCMRA 3
```

```
#define IMAGE_NONE_NONE 0
```

```
#define IMAGE_CT_NONE 1
#define IMAGE_CT_HEAD 2
#define IMAGE_CT_LUNG 3
```

```
#define IMAGE_MR_NONE 1
#define IMAGE_MR_HEAD 2
#define IMAGE_MR_LUNG 3
```

```
#define IMAGE_PCMRA_NONE 1
#define IMAGE_PCMRA_HEAD 2
#define IMAGE_PCMRA_LUNG 3
```

```
#define PCMRA_MAGNITUDE 0
#define PCMRA_PHASE 1
#define PCMRA_VELOCITY 2
```

```
#define IMAGE_2D 0
#define IMAGE_3D 1
```

```
#define LAYOUT_NORMAL 0
#define LAYOUT_COMBO 1
```

```
#define INTERPOLATION_SIMPLE 0
#define INTERPOLATION_SPLINE 1
```

```
#define RIGHT_IMG_WHOLE 0
#define RIGHT_IMG_ROI 1
#define RIGHT_IMG_OTHER 2
#define RIGHT_IMG_REF 3
```

```
#define VISUAL_2D_GRAY 0
#define VISUAL_2D_COLOR 1
```

```
#define HISTOGRAM_COARSE 0
#define HISTOGRAM_FINE 1
#define HISTOGRAM_ROI 2
#define HISTOGRAM_MAPPING 3
```

```
#define ZOOM_LEFT 0
```



```
#define ZOOM_RIGHT 1
#define ZOOM_BOTH 2
```

606

```
#define FLOW_VFR 0
#define FLOW_PSV 1
#define FLOW_BSV 3
#define FLOW_MV 4
#define FLOW_AREA 5
```

```
#define ANIMATE_L1D 0
#define ANIMATE_R1D 1
#define ANIMATE_1D 2
#define ANIMATE_2D 3
#define ANIMATE_3D 4
#define ANIMATE_SYMPHONY 5
```

```
#define FLOW_MANUAL 0
#define FLOW_SEMIAUTO 1
#define FLOW_AUTOSNAKE 2
```

```
#define VELOCITY_ASIS 0
#define VELOCITY_AUTO 1
#define VELOCITY_ROIMASKED 2
#define VELOCITY_FLOWMASKED 3
```

```
#define CAMERA_ORTHO 0
#define CAMERA_PERSPECTIVE 1
#define CAMERA_PERSPECTIVE_ROT 2
```

```
#define USER_NOVIES 0
#define USER_EXPERT 1
```

```
#define FLOW3D_DISABLE 0
#define FLOW3D_ENABLE 1
```

```
#define PUBLISH_NONE 0
#define PUBLISH_2DMAG 1
#define PUBLISH_2DPHA 2
#define PUBLISH_2DLOC 3
#define PUBLISH_2DWAVE 4
#define PUBLISH_3DLOC 5
#define PUBLISH_3DFLOW 6
```

```
class Utility
{
```

```
public:
```

```
Utility ();
~Utility();
```

```
int get_ImgType(char *type);
int get_ImgAnatomy(int type);
int get_ImgAnatomy(int img_type, char *anatomy);
void get_GE(int img_type, int img_level,
float *widCenter, float *winWidth);
```

```
GE_PCMRA_HEADER_OBJ *copy_pc(GE_PCMRA_HEADER_OBJ *);
```

```
FlowPara *get_flow(int w, int h, short **img, float pixel_area,
unsigned char **mask, unsigned char **back = NULL);
```

```
short **ToVelocity(GE_PCMRA_HEADER_OBJ *pc, int w, int h,
short **mag_img, short **pha_img, float posThresh = 0.0,
```

```
float negThresh = 0.0, float magThresh = 0.0);
short **ToVelocityROI[10][10] = {0};
short **mag_img,
short **pha_img, int x, int y, int w, int h, unsigned char **mask,
float posThresh = 0.0, float negThresh = 0.0, float magThresh = 0.0);

void GE_RAS_CenterNormal2Points(GE_PCMRA_HEADER_OBJ *pc_loc,
GE_PCMRA_HEADER_OBJ *pc_phase, int *xx1, int *yy1, int *xx2, int *yy2);

void get_point(int num, Point *point, float *x, float *y);
//
// find the center point of the "Point *point"
//

void get_point(float xc, float yc, float sita, int num,
Point *point, float *x, float *y);
//
// find the point in the boundary that is made of the "Point *point"
// so that the angle of the line connecting the point and the center
// point (xc,yc) is "sita"
//

float get_angle(float x1, float y1, float x2, float y2);
//
// find the angle of the line connecting the given two points
//

private:

};

#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: Utility_Vision.h

608

```
#ifndef IMAGEALLOC_H
#define IMAGEALLOC_H
```

609

```
unsigned char **alloc_img (int xsize,int ysize);
void free_img (unsigned char **img );
```

```
float **alloc_fimg (int xsize,int ysize);
void free_fimg (float **img );
```

```
short **alloc_shimg (int xsize,int ysize);
void free_shimg (short **img );
```

```
unsigned long **alloc_ulimg (int xsize,int ysize);
void free_ulimg (unsigned long **img );
```

```
#endif
```

```
#ifndef GE_CT_MRI_H
#define GE_CT_MRI_H
```

610

```
#include <stdlib.h>
#include <stdio.h>
```

```
#define IMG_HEADER_SIZE 156
#define MR_HEADER_SIZE 1022
#define CT_HEADER_SIZE 1020
```

```
typedef struct {
    FILE      *fp;
    fpos_t    filePosition;
    int       numberOfBytesRead;
    short     **img;
} CANVAS_OBJ;
```

```
typedef struct {
    int img_width;
    int img_height;

    short hs_min;
    short hs_max;

    float slthick; /* The slice thickness */
    short imatrix_X; /* Image Matrix size - X */
    short imatrix_Y; /* Image Matrix size - Y */
    float dfov; /* Display field of View - X(mm) */

    float dfov_rect; /* Display field of view - Y(if different) */
    float dim_X; /* Image Dimension - X */
    float dim_Y; /* Image Dimension - Y */

    float pixsize_X; /* Image X-Pixel Size */
    float pixsize_Y; /* Image Y-Pixel Size */
    float scanspacing; /* Inter-Slice spacing */
    int tr;
    int te;
    float num_excitations;
    short heart_rate;
    int delay_time;
    short num_img_per_cardiac_cycle;
    short flip_angle;
    short pc_flow_axis;
    short pc_venc;
    short cardiac_phase_num;
    int num_of_phases;
    float min_I,max_I;
    int mag_weighting_flag;
    float venc_weighted_scale;

    char loc_ras;
    float ctr_R, ctr_A, ctr_S;
    float norm_R, norm_A, norm_S;
    float tlh_R, tlh_A, tlh_S;
    float trh_R, trh_A, trh_S;
    float brh_R, brh_A, brh_S;
```

```
} GE_PCMRA_HEADER_OBJ;
```

```
typedef struct {
    int img_magic; /* image file unique magic number */
    int img_hdr_length; /* length of header, also byte displacement
                        * to the pixel data area
                        */
```

```

int img_width;      /* x-axis pixel count */
int img_height;     /* y-axis pixel count */

int img_depth;      /* number of bits in an uncompressed pixel
 * (1, 8, 16, or 24). NOTE: NOT magnitude resolution
 * (CT is 16, not 12)
 */

int img_compress;   /* Form of compression and packing applied to file,
 * where:
 * 1 = IC_RECT Non-compressed, normal rect'lar img
 * 2 = IC_PACKED Img is line length map packed
 * 3 = IC_COMPRESSED Img is compressed via DCPM only
 * 4 = IC_COMPACK Img is compressed and packed.
 */

int img_dwindow;    /* default window width (stored image value range) */
int img_dlevel;     /* default level value (stored image value magnitude) */
int img_bgshade;    /* default bkgrnd shade for non-pixels during unpack */
int img_ovrflow;    /* pix val to subs when overflow occurs in GIP */
int img_undflow;    /* pix val to subs when underflow occurs in GIP */
int img_top_offset; /* num of (blank) lines at top of the img */
int img_bot_offset; /* num of (blank) lines at bot of the img */
short img_version;  /* version of the header structure IMG_HDR_VERSION */
unsigned short img_checksum; /* 16 bit end_around_carry sum of true
 * img pixels. A val of 0 indicates the
 * checksum is not defined for this file
 */

int img_p_id;       /* a byte disp to 'unique img identifier text table' */
int img_l_id;       /* byte length of 'unique img identifier text table' */
int img_p_unpack;   /* a byte disp to 'unpack control table' */
int img_l_unpack;   /* byte length of 'unpack control table' */
int img_p_compress; /* a byte disp to 'compression seed table' */
int img_l_compress; /* byte length of 'compression seed table' */
int img_p_histo;    /* a byte disp to 'histogram control table' */
int img_l_histo;    /* byte length of 'histogram control table' */
int img_p_text;     /* a byte disp to 'text plane data' */
int img_l_text;     /* byte length of 'text plane data' */
int img_p_graphics; /* a byte disp to 'graphics plane data' */
int img_l_graphics; /* byte length of 'graphics plane data' */
int img_p_dbHdr;    /* a byte disp to 'database header data' */
int img_l_dbHdr;    /* byte length of 'database header data' */
int img_levelOffset; /* val to add to actual pix data vals to get the correct
 * annotation val. For CT, physical 0 means - 1024
 * Hounsfield number.
 */

int img_p_user;     /* a byte disp to 'user defined data' */
int img_l_user;     /* byte length of 'user defined data' */
int img_p_suite;    /* a byte disp to 'suite header data' */
int img_l_suite;    /* byte length of 'suite header data' */
int img_p_exam;    /* a byte disp to 'exam header data' */
int img_l_exam;    /* byte length of 'exam header data' */
int img_p_series;   /* a byte disp to 'series header data' */
int img_l_series;   /* byte length of 'series header data' */
int img_p_image;    /* a byte disp to 'series image data' */
int img_l_image;    /* byte length of 'series image data' */
} GE_HEADER_OBJ;

typedef struct {
    int hs_version;
    float hs_sd;
    short hs_mean;

    short hs_min;
    short hs_max;

    short hs_first;

```

```

short hs_region;
short hs_length;
unsigned short hs_bins[24];
} GE_HISTO_HEADER_OBJ;

```

```

typedef struct {
    short im_no;      /* The Image Number for this Image */

    float slthick;    /* The slice thickness */
    short imatrix_X;  /* Image Matrix size - X */
    short imatrix_Y;  /* Image Matrix size - Y */
    float dfov;       /* Display field of View - X(mm) */
    float pixsize_X;  /* Image X-Pixel Size */
    float pixsize_Y;  /* Image Y-Pixel Size */
    float scanspacing; /* Inter-Slice spacing */

    float dfov_rect;  /* Display field of view - Y(if different) */
    float dim_X;       /* Image Dimension - X */
    float dim_Y;       /* Image Dimension - Y */
    short planes;     /* Number of planes */
} GE_CT_HEADER_OBJ;

```

```

typedef struct {
    short im_no;      /* The Image Number for this Image */

    float slthick;    /* The slice thickness */
    short imatrix_X;  /* Image Matrix size - X */
    short imatrix_Y;  /* Image Matrix size - Y */
    float dfov;       /* Display field of View - X(mm) */

    float dfov_rect;  /* Display field of view - Y(if different) */
    float dim_X;       /* Image Dimension - X */
    float dim_Y;       /* Image Dimension - Y */

    float pixsize_X;  /* Image X-Pixel Size */
    float pixsize_Y;  /* Image Y-Pixel Size */
    float scanspacing; /* Inter-Slice spacing */

```

```

    short planes; /* Number of planes */

```

```

/* Meide's Stuff Begin */
short img_compress; /* image compression type */
short img_scouttype; /* Scout type */
char loc_ras; /* RAS letter of image location */
char tmp;

```

```

float loc; /* image location */

```

```

int tr; /* pulse repetition */
int te; /* pulse echo */

```

```

int ti; /* pulse inversion */
int te2; /* second echo echo */

```

```

short num_echo;
short echo_num;
float table_delta;
short continuous_slice_flag;
float cardiac_rep_time;

```

```

float num_excitations;
short heart_rate;
int delay_time;
float saravg;
float sarpeak;

```

```
short  monsar;
short  trgwindow;
float  reptime;
int    min_delay_time;

short  num_img_per_cardiac_cycle;
short  flip_angle;
short  pc_flow_axis;
short  pc_venc;
short  cardiac_phase_num;
int    num_of_phases;

short  img_type;
char   tmp_array[1000];
char   *foreign_img;

short  scan_acquisition_no;
short  mag_wighting_flag;
float  venc_weighted_scale;

float  ctr_R, ctr_A, ctr_S;
float  norm_R, norm_A, norm_S;
float  tlh_R, tlh_A, tlh_S;
float  trh_R, trh_A, trh_S;
float  brh_R, brh_A, brh_S;
/* Meide's Stuff End */
} GE_MR_HEADER_OBJ;

short **read_GE_CT_MRI(char *,GE_PCMRA_HEADER_OBJ *);
void read_GE(char *,GE_PCMRA_HEADER_OBJ *, short **);

#endif
```



```
#ifndef IMGBASE_H
#define IMGBASE_H
```

614

```
class ObjectManager;
```

```
class ImgBase
{
```

```
public:
```

```
    ImgBase ();
    ImgBase (int, int, short **);
    ~ImgBase();

    int get_width() {return width;}
    int get_height(){return height;}
    short **get_imgdata();

    void set_width(int w) {width = w;}
    void set_height(int h) {height = h;}
    void set_imgdata(short **img);
```

```
protected:
```

```
    short **imgdata;
    int    width;
    int    height;
```

```
};
```

```
#endif
```

```
#ifndef IMGGE_H
#define IMGGE_H
```

615

```
#include "GE.h"
#include <Vk/VkComponent.h>
#include "ImgBase.h"
```

```
class ImgGE : public ImgBase
{
```

```
public:
```

```
    ImgGE (char *filename);
    ImgGE ();
    ~ImgGE();
```

```
    ImgGE *copy();
```

```
    void set(char *fname);
    void set(ImgGE *);
    void set(unsigned char **mask, float in_ratio);
    void set(float zoom, short **img, int x, int y, int w, int h);
    void set(int x, int y, int w, int h, short **img);
    void set(int x, int y, int w, int h, short **img, unsigned char **mask, float ratio);
    unsigned char **thresh(int x, int y, int w, int h, float low) ;
```

```
    void inverseImg();
    short **get_ROI(int x1, int y1, int w1, int h1, short **img);
    short **get_ROI(int x1, int y1, int w1, int h1, short **img,
        unsigned char **mask, float ratio);
```

```
    float get_pixel_area() {return (pc->pixsize_X * pc->pixsize_Y / 100.0);}
    float get_pixelX() {return pc->pixsize_X;}
    float get_pixelY() {return pc->pixsize_Y;}
    short get_heart_rate() {return pc -> heart_rate;}
    GE_PCMRA_HEADER_OBJ *get_header();
```

```
    void set_header(GE_PCMRA_HEADER_OBJ *in_pc) {pc = in_pc;}
```

```
private:
```

```
    GE_PCMRA_HEADER_OBJ *pc;
```

```
};
```

```
#endif
```

```
#ifndef OBJECTMANAGER_H
#define OBJECTMANAGER_H
```

616

```
//---- Start editable code block: headers and declarations
```

```
#include <Vk/VkComponent.h>
#include <Vk/VkSimpleWindow.h>
```

```
#include <stdio.h>
#include "MessagesLoaded.h"
#include "MessagesLeft.h"
#include "MessagesRight.h"
```

```
#include "ROIS.h"
```

```
#include "ImgGE.h"
#include "ROI MedDrawingArea.h"
```

```
#include "BbHistogram.h"
#include "BbRHistogram.h"
```

```
#include "Patients.h"
#include "GE.h"
```

```
#include <Inventor/nodes/SoSeparator.h>
```

```
//---- End editable code block: headers and declarations
```

```
//---- ObjectManager class declaration
```

```
class ObjectManager
{
```

```
    public:
```

```
    ObjectManager ( );
    ~ObjectManager();
```

```
    void init();
```

```
    //---- Start editable code block: ObjectManager public
```

```
    Patients          *_patients;
    MessagesLoaded    msgsLoaded;
    MessagesLeft      msgsLeft;
    MessagesRight     msgsRight;
    ROIS              *_ROIS;
```

```
    int               _vessel, _num_vessels;
    Flow              *_flow;
```

```
    SoSeparator       *_root;
    GE_PCMRA_HEADER_OBJ *_GE_header;
```

```
    void get_general();
```

```
    ImgGE              *_img;
    class ROI MedDrawingArea *_imgView;
    void set_Img(class ImgGE *img) {_img = img;}
    void set_ImgView(class ROI MedDrawingArea *imgView) {_imgView = imgView;}

    ImgGE              *_img2;
    class ROI MedDrawingArea *_imgView2;
    void set_Img2(class ImgGE *img) {_img2 = img;}
    void set_ImgView2(class ROI MedDrawingArea *imgView) {_imgView2 = imgView;}
```

```
class ROI MedDrawingArea *_imgViewLoc;
```

617

```
class HistoTwoLinesDrawingArea *_histoView;  
class HistoTwoLinesDrawingArea *_histoView2;  
void set_HistoView(class HistoTwoLinesDrawingArea *v) { _histoView= v;}  
void set_HistoView2(class HistoTwoLinesDrawingArea *v) { _histoView2= v;}
```

```
DrawingAreaUI *_waveView;  
DrawingAreaUI *_waveView2;
```

```
class MedDrawingArea *_map;  
ImgGE *_magImg;  
ImgGE *_phaImg;
```

```
class BbUI *_bb;  
void set(class BbUI *bb) {_bb = bb;}
```

```
class DeckLTabbedDeck *_deckL;  
class DeckRTabbedDeck *_deckR;
```

```
class BbDisplay *_LDisp;  
class BbLROI *_LROI;  
class BbHistogram *_LHist;  
class BbDetail *_LDet1;  
class BbLPCMRA *_LPCMRA;  
class BbLWaveform *_LWave;  
class BbLConfig *_LConfig;
```

```
void set(class BbDisplay *b) {_LDisp = b;}  
void set(class BbLROI *b) {_LROI = b;}  
void set(class BbHistogram *b) {_LHist = b;}  
void set(class BbDetail *b) {_LDet1 = b;}  
void set(class BbLPCMRA *b) {_LPCMRA = b;}  
void set(class BbLWaveform *b) {_LWave = b;}  
void set(class BbLConfig *b) {_LConfig = b;}
```

```
class BbVisual *_RVis1;  
class BbRROI *_RROI;  
class BbRHistogram *_RHist;  
class BbFlow *_RFlow;  
class BbRWaveform *_RWave;  
class BbRTable *_RTable;  
class BbAnimation *_RAnimate;  
class Bb3D *_b3D;  
class BbVelocity *_bVelocity;  
class BbFlow3D *_flow3d;  
class Bb3DLocalizer *_localizer3d;  
class BbFormat *_RFormat;
```

```
void set(class BbVisual *b) {_RVis1 = b;}  
void set(class BbRROI *b) {_RROI = b;}  
void set(class BbRHistogram *b) {_RHist = b;}  
void set(class BbFlow *b) {_RFlow = b;}  
void set(class BbRWaveform *b) {_RWave = b;}  
void set(class BbRTable *b) {_RTable = b;}  
void set(class BbAnimation *b) {_RAnimate = b;}  
void set(class Bb3D *b) {_b3D = b;}  
void set(class BbVelocity *b) {_bVelocity = b;}  
void set(class BbFlow3D *b) {_flow3d = b;}  
void set(class Bb3DLocalizer *b) {_localizer3d = b;}  
void set(class BbFormat *b) {_RFormat = b;}
```

```
void update_Aimg(int img_number);  
//void update_AimgView(float zoom);
```

```

void update_Limg(int img_number);
void update_LimgView();
void update_LimgView2D();
Boolean get_LscaleSize(float zoom, int *w, int *h);
void new_LimgView();
void update_Lhisto();
void update_Lhisto2();

void update_Rimg(int img_number);
void update_RimgView();
void update_RimgView2D();
void update_Rimg2D();
Boolean get_RscaleSize(float zoom, int *w, int *h);
void new_RimgView();
void new_RimgViewLoc();
float get_Rzoom(int w, int h);
void update_Rhisto();
void update_Rhisto1();
void update_Rhisto2();
void update_RhistoROI();
void update_RhistoMapping();

void update_Rimg2D(ImgGE *);
void update_mask(ImgGE *);

void update_Lwave(int vessel = 0);
void update_Rwave(int vessel = 0);

ImgGE *get_ImgGE(int, int, int img_number, ImgGE *imgGE);
ImgGE *get_ImgGE2(int img_number, ImgGE *ie);

void create_animate();
void remove_animate();
void create_Lanimate1D();
void create_Ranimate1D();
void create_animate2D();
void start_animate();

void create_iv(int, int, short **);

void update_flow();
void saveFlow();
void get_minmaxFlow(int flag = 0);

void localizer();

void update_LimgView(float center, float width);

void update_RimgView(float center, float width);
void update_RimgView(int scale_method);
void update_RimgView(float zoom);
void update_Rvisual(int visual_method);

void set_Llowhigh();
void set_Rlowhigh();
void update_Llowhigh();
void update_Rlowhigh();
void update_mask();

int get_tag(int num, float *x, float *xMin, float *xMax, float *avg);

void remove_progress();
void update_progress(char *);

void display_ROI();

```

```
void show2D();
void hide2D();
```

```
void showL2D();
void hideL2D();
```

```
class SoXtExaminerViewer    *_L3D;
class SoXtExaminerViewer    *_R3D;
```

```
void set_ratio3D();
void hide3D();
void hideL3D();
```

```
void empty_animate3D();
void remove_animate3D();
void create_Ranimate3D();
void create_animateSymphony();
```

```
void update_RimgView3D();
void update_RimgView3DROI();
void update_LimgView3D();
```

```
VkSimpleWindow    *_win3D;
void update_win3D();
```

```
//---- End editable code block: ObjectManager public
```

```
protected:
```

```
//---- Start editable code block: ObjectManager protected
```

```
//---- End editable code block: ObjectManager protected
```

```
private:
```

```
//---- Start editable code block: ObjectManager private
```

```
//---- End editable code block: ObjectManager private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```
#ifndef READCONFIG
#define READCONFIG
```

620

```
#include "MessagesLoaded.h"
#include "MessagesLeft.h"
#include "MessagesRight.h"
```

```
MessagesLoaded ReadConfig();
MessagesLeft ReadConfigLeft(int img_type);
MessagesRight ReadConfigRight(int img_type);
```

```
#endif
```

```
#ifndef ELLIPSE_H
#define ELLIPSE_H
```

621

```
#include "ROI.h"
#include "Points.h"
```

```
#define LEFT_TOP      0
#define LEFT_BOTTOM   1
#define RIGHT_TOP      2
#define RIGHT_BOTTOM   3
```

```
class Ellipse : public ROI
{
```

```
public:
```

```
    Ellipse (Widget w, int);
    ~Ellipse();
```

```
    virtual void init(int x, int y);
    virtual void new_started(int x, int y);
    virtual void motion(int x, int y);
    virtual void released(int, int);
    virtual void finished(int, int) {AcceptROI();}
```

```
    virtual void motion_modify(int x, int y);
    virtual void init_modify(int, int);
    virtual void released_modify(int, int) {}
```

```
    virtual void motion_move(int x, int y);
    virtual void init_move(int, int);
```

```
    virtual void get_BoundingBox(int *x, int *y, int *w, int *h);
    virtual void draw();
    virtual void fill();
```

```
    int  _xlen;
    int  _ylen;
    float _angle;
    int  _corner;
```

```
protected:
```

```
    int get_points_in_border();
    void transform(float x1, float y1, float *x, float *y);
```

```
private:
```

```
};
```

```
#endif
```



```
#ifndef RECTANGLE_H
#define RECTANGLE_H
```

622

```
#include "ROI.h"
```

```
#define LEFT_TOP 0
#define LEFT_BOTTOM 1
#define RIGHT_TOP 2
#define RIGHT_BOTTOM 3
```

```
class Rectangle : public ROI
{
```

```
public:
```

```
    Rectangle(Widget w, int);
    ~Rectangle();
```

```
    int _x, _y, _w, _h;
```

```
    virtual void init(int x, int y);
    virtual void new_started(int x, int y);
    virtual void motion(int x, int y);
    virtual void released(int, int);
    virtual void finished(int, int) {_draw_status = TRUE; AcceptROI();}
```

```
    virtual void motion_modify(int x, int y);
    virtual void init_modify(int, int);
    virtual void released_modify(int, int) {}
```

```
    virtual void motion_move(int x, int y);
    virtual void init_move(int, int);
```

```
    virtual void get_BoundingBox(int *x, int *y, int *w, int *h);
    virtual void draw();
    virtual void fill();
```

```
    int _corner;
```

```
protected:
```

```
    void get_points_in_border();
```

```
private:
```

```
};
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: ImgAlloc.h

623

```
#ifndef POLYGON_H
#define POLYGON_H
```

624

```
#include "ROI.h"
#include "Points.h"
```

```
class Polygon : public ROI
{
```

```
    public:
```

```
        Polygon(Widget w, int);
        ~Polygon();
```

```
        virtual void init(int x, int y);
        virtual void new_started(int x, int y);
        virtual void motion(int x, int y);
        virtual void released(int, int);
        virtual void finished(int, int);
```

```
        virtual void motion_modify(int x, int y);
        virtual void init_modify(int, int);
        virtual void released_modify(int, int);
```

```
        virtual void motion_move(int x, int y);
        virtual void init_move(int, int);
```

```
        virtual void get_BoundingBox(int *x, int *y, int *w, int *h);
        virtual void draw();
        virtual void fill();
```

```
        int _x, _y;
```

```
        int _modify_num;
```

```
    protected:
```

```
    private:
```

```
};
```

```
#endif
```

```

#ifndef FREEHAND_H
#define FREEHAND_H

#include "ROI.h"
#include "Points.h"

#define NUM_POINTS 25

class FreeHand : public ROI
{
public:
    FreeHand(Widget w, int);
    ~FreeHand();

    virtual void init(int x, int y);
    virtual void new_started(int x, int y);
    virtual void motion(int x, int y);
    virtual void released(int, int);
    virtual void finished(int, int);

    virtual void motion_modify(int x, int y);
    virtual void init_modify(int, int);
    virtual void released_modify(int, int);

    virtual void motion_move(int x, int y);
    virtual void init_move(int, int);

    virtual void get_BoundingBox(int *x, int *y, int *w, int *h);
    virtual void draw();
    virtual void fill();

    void init_modify2(int);
    int scaling(int, int);

    int _x, _y;

    int _numPoints1;
    int _numPoints2;
    int _tmpPN1[NUM_POINTS];
    int _tmpPN2[NUM_POINTS];
    Point _tmpPoints1[NUM_POINTS];
    Point _tmpPoints2[NUM_POINTS];

    int _modify_num;

protected:

private:
};

#endif

```

```

#ifndef POINTS_H
#define POINTS_H

#include <Vk/VkComponent.h>
#include <stdio.h>
#include "Point.h"
#include "Utility_Vision.h"

class Points
{
public:
    Points ();
    ~Points();

    Points *create();

    void clear();

    void init(int x, int y);
    void add(int x, int y);
    void add(float x, float y);
    void show_info();
    XPoint *get_XPoint();

    void draw(Widget, GC);
    void draw(Widget, int color);
    void draw_noloop(Widget, GC);
    void draw_keyPoints(Widget);

    void draw_line(Widget w, Point p1, Point p2, int w1, int h1, ColorImage *cimg);
    void draw_img(Widget w, int w1, int h1, ColorImage *cimg);

    void translation1(int xc, int yc);
    void translation2(int dx, int dy);

    Points *get_Points(float zoom, int x, int y);
    Points *inverse_get_Points(float zoom, int x, int y);
    void zoom(float zoom, int x, int y);
    void inverse_zoom(float zoom, int x, int y);

    int get_minmax(float *min_x, float *max_x, float *min_y, float *max_y);

    int get_points_in_between(Point p1, Point p2, Point *point);
    int neighbor(Point p1, Point p2);

    void fill();
    int closest(int x, int y);

    void to_File(FILE *fp);
    void to_ContourFile(FILE *fp);
    void from_ContourFile(FILE *fp);
    void to_File(int k, FILE *fp, float thickness, float pX, float pY);
    void to_ivFile(int k, FILE *fp, float thickness, float pX, float pY, int index=0);
    void to_ivFileModify(int k, FILE *fp, float thickness, float pX, float pY, int inc);
    void from_File(FILE *fp);

    int _numPoints;
    int _currPoints;
    Point _points[10000];

private:
};

```

#endif

```
#ifndef ROI_H
#define ROI_H
```

626

```
#include <Vk/VkComponent.h>
#include "Points.h"
#include "Utility_Widget.h"
#include "Utility_Vision.h"
```

```
#define ROI_NONE          0
#define ROI_RECTANGLE    1
#define ROI_ELLIPSE      2
#define ROI_FREEHAND     3
#define ROI_POLYGON      4
```

```
#define ROI_REDEFINE      1
#define ROI_REDEFINE_NO  2
#define ROI_MODIFY        3
#define ROI_MOVE          4
#define ROI_SCALE         5
```

```
class ROI
{
```

```
    public:
```

```
    ROI (Widget w, int color=COLOR_RED);
    ~ROI();
```

```
    void set_area();
    unsigned char **copyArea();
    void set_areaOrg(float zoom);
    void draw_img();
    void AcceptROI();
    Boolean inArea(int x, int y) { return ((_area[y][x] == 1) ? TRUE : FALSE); }
```

```
    virtual void init(int x, int y) = 0;
    virtual void new_started(int x, int y) = 0;
    virtual void motion(int x, int y) = 0;
    virtual void released(int, int) = 0;
    virtual void finished(int, int) = 0;
```

```
    virtual void motion_modify(int x, int y) = 0;
    virtual void init_modify(int, int) = 0;
    virtual void released_modify(int, int) = 0;
```

```
    virtual void motion_move(int x, int y) = 0;
    virtual void init_move(int, int) = 0;
```

```
    virtual void get_BoundingBox(int *x, int *y, int *w, int *h) = 0;
    virtual void draw() = 0;
    virtual void fill() = 0;
```

```
//
//    Data Members
//
```

```
    Widget _widget;
    GC      _gc;
```

```
    Point  _start;
    Point  _center;
```

```
    int    _width;
    int    _height;
```

```
    class ROIMedDrawingArea *_roiView;
```

```
int    _event;

Boolean _draw_status;
Boolean _show_status;

unsigned char **_area;
unsigned char **_areaOrg;

Points  _points_in_border;

protected:

private:

};

#endif
```



```
#ifndef ROIS_H
#define ROIS_H
```

628

```
#include <Vk/VkComponent.h>
#include "ROI_Struct.h"
```

```
class ROIS
{
```

```
    public:
```

```
        ROIS (int );
        ~ROIS();
```

```
        void add(int img_number, char *name, Points *p);
```

```
        void remove(int img_number, int roi_number);
```

```
        void to_File();
```

```
        void to_File(float thickness, float pX, float pY);
```

```
        void to_ivFile(float thickness, float pX, float pY);
```

```
        void to_ivFile(int, int, float thickness, float pX, float pY);
```

```
        void to_ivFileSurface(float thickness, float pX, float pY);
```

```
        void from_File(char *fname) ;
```

```
        int get_index(char *name);
```

```
        int          _numFrames;
        ROI_Struct    *_ROI;
```

```
    private:
```

```
};
```

```
..#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: Polygon.h

629

```
#ifndef TWOLINES_H
#define TWOLINES_H
```

630

```
#include <Vk/VkComponent.h>
```

```
#define TWOLINES_MOVE 1
#define TWOLINES_LEFT 2
#define TWOLINES_RIGHT 3
#define TWOLINES_INCREASE_LEFT 4
#define TWOLINES_INCREASE_RIGHT 5
```

```
class TwoLines
```

```
{
```

```
public:
```

```
TwoLines(Widget w, int, int);
~TwoLines();
```

```
int _status;
Widget _widget;
GC _gc;
```

```
float _current_x;
float _x1, _x2;
int _max_x, _max_y;
```

```
void draw();
void init(int x);
void draw(int x);
```

```
void set(int x1, int x2) {_x1 = x1; _x2 = x2;}
void set(float center, float width);
```

```
protected:
```

```
private:
```

```
};
```

```
#endif
```

```
#ifndef ANIMATE_H
#define ANIMATE_H
```

631

```
#include <Vk/VkComponent.h>
#include "DrawingAreaUI.h"
#include "Utility_3D.h"
```

```
typedef struct {
    SoSeparator      *_iv;
} SOIV;
```

```
typedef struct {

    Boolean          _toBeFinished;

    int              _num_imgs;
    Widget           _widget;

    int              _msec;
    int              _img_number;
    int              _time_out;
    int              _firsttime;
    XtIntervalId     _id;
```

```
    int      _width;
    int      _height;
    GC       _gc;
    Pixmap   *_pixmap;
```

```
    int      _wave_number;
    int      _num_waves;
```

```
    DrawingAreaUI *_lWave;
    int           _lWaveColor;
    GC            _lWaveGC;
```

```
    DrawingAreaUI *_rWave;
    int           _rWaveColor;
    GC            _rWaveGC;
```

```
    SoXtExaminerViewer *_ivview;
    SOIV                *_soiv;
```

```
} Animate;
```

```
Animate  *_animate;
```

```
void animation();
```

```
#endif
```

```
#ifndef PROGRESS_H
#define PROGRESS_H
```

632

```
#include <Vk/VkComponent.h>
#include "ObjectManager.h"
#include "ProgressMainWindow.h"
```

```
typedef struct {
```

```
    ObjectManager *_objMag;
    Widget        widget;

    ProgressMainWindow *window;
```

```
    int           curr;
```

```
    int           msec;
```

```
    int           time_out;
```

```
    int           firsttime;
```

```
    XtIntervalId  id;
```

```
} Progress;
```

```
Progress *progress;
```

```
void Progress_Animate2D();
```

```
void Progress_Animate3D();
```

```
void Progress_AnimateSymphony();
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for SwUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, Sw is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the Sw files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef SWUI_H
#define SWUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

#include <Xm/Text.h>

/*--- End editable code block: headers and declarations

class SwUI : public VkComponent
{
    public:

        SwUI ( const char *, Widget );
        SwUI ( const char * );
        ~SwUI();
        void create ( Widget );
        const char * className();

        /*--- Start editable code block: Sw public

        void set(char *str) { XmTextSetString(_scrolledText, str); }

        /*--- End editable code block: Sw public

    protected:

        // Widgets created by this class

        Widget _scrolledText;
        Widget _sw;

        /*--- Start editable code block: Sw protected

```

//---- End editable block: Sw protected

634

private:

// Array of default resources

static String _defaultSwUIResources[];

//---- Start editable code block: Sw private

//---- End editable code block: Sw private

};

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```

////////////////////////////////////
//
// Header file for Sw
//
// This file is generated by RapidApp 1.2
//
// This class is derived from SwUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef SW_H
#define SW_H
#include "SwUI.h"
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- Sw class declaration

class Sw : public SwUI
{
public:
    Sw ( const char *, Widget );
    Sw ( const char * );
    ~Sw();
    const char * className();

    static VkComponent *CreateSw( const char *name, Widget parent );

    //---- Start editable code block: Sw public

    //---- End editable code block: Sw public

protected:

    //---- Start editable code block: Sw protected

    //---- End editable code block: Sw protected

private:
    static void* RegisterSwInterface();

    //---- Start editable code block: Sw private

```



```
//---- End editable code block: Sw private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for InfoMainWindow
//
// This class is a subclass of VkSimpleWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
////////////////////////////////////
#ifdef INFOMAINWINDOW_H
#define INFOMAINWINDOW_H
#include <Vk/VkSimpleWindow.h>

//--- Start editable code block: headers and declarations

#include "Sw.h"

//--- End editable code block: headers and declarations

//--- InfoMainWindow class declaration

class InfoMainWindow: public VkSimpleWindow {

public:

    InfoMainWindow( const char * name,
                    ArgList args = NULL,
                    Cardinal argCount = 0 );
    ~InfoMainWindow();
    const char *className();
    virtual Boolean okToQuit();

    //--- Start editable code block: InfoMainWindow public

    void set(char *str) { ((Sw *)_sw) -> set(str); }

    //--- End editable code block: InfoMainWindow public

protected:

    // Classes created by this class

    class Sw *_sw;

    // Widgets created by this class

    //--- Start editable code block: InfoMainWindow protected

    //--- End editable code block: InfoMainWindow protected

```

private:

638

```
static String _defaultInfoMainWindowResources[];
```

```
//---- Start editable code block: InfoMainWindow private
```

```
//---- End editable code block: InfoMainWindow private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for ProgressBB
//
// This file is generated by RapidApp 1.2
//
// This class is derived from ProgressBBUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef PROGRESSBB_H
#define PROGRESSBB_H
#include "ProgressBBUI.h"
//---- Start editable code block: headers and declarations

#include "MedDrawingArea.h"

//---- End editable code block: headers and declarations

//---- ProgressBB class declaration

class ProgressBB : public ProgressBBUI
{
public:
    ProgressBB ( const char *, Widget );
    ProgressBB ( const char * );
    ~ProgressBB();
    const char * className();

    static VkComponent *CreateProgressBB( const char *name, Widget parent );

//---- Start editable code block: ProgressBB public

    int          _width, _height;
    short        **_mapImg;
    MedDrawingArea *_map;

    int          _percent;
    Boolean       _cancel;

    void init(char *msg);
    void set_title(char *msg);
    void set_percent(int);

    void update_percent(int curr, int num);

//---- End editable code block: ProgressBB public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in ProgressBBUI

virtual void doButtonCancel ( Widget, XtPointer );

//---- Start editable code block: ProgressBB protected

//---- End editable code block: ProgressBB protected

private:

    static void* RegisterProgressBBInterface();

    //---- Start editable code block: ProgressBB private

    //---- End editable code block: ProgressBB private

};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif
```

```

////////////////////////////////////
//
// Header file for ProgressBBUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, ProgressBB is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the ProgressBB files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef PROGRESSBBUI_H
#define PROGRESSBBUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class ProgressBBUI : public VkComponent
{
public:

    ProgressBBUI ( const char *, Widget );
    ProgressBBUI ( const char * );
    ~ProgressBBUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: ProgressBB public

    //---- End editable code block: ProgressBB public

protected:

    // Widgets created by this class

    Widget _buttonCancel;
    Widget _frame;
    Widget _labelPercent;
    Widget _labelTitle;
    Widget _progressBB;

    // These virtual functions are called from the private callbacks (below)

```

```
// Intended to be overriden in derived classes to define actions
virtual void doButtonCancel ( Widget, XtPointer );

//----- Start editable code block: ProgressBB protected

//----- End editable code block: ProgressBB protected

private:

// Array of default resources
static String      _defaultProgressBBUIResources[];

// Callbacks to interface with Motif
static void doButtonCancelCallback ( Widget, XtPointer, XtPointer );

//----- Start editable code block: ProgressBB private

//----- End editable code block: ProgressBB private
};
//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: TwoLines.h

643


```

////////////////////////////////////
//
// Header file for ProgressMainWindow
//
// This class is a subclass of VkSimpleWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
////////////////////////////////////
#ifndef PROGRESSMAINWINDOW_H
#define PROGRESSMAINWINDOW_H
#include <Vk/VkSimpleWindow.h>

/*--- Start editable code block: headers and declarations

#include "ProgressBB.h"

/*--- End editable code block: headers and declarations

/*--- ProgressMainWindow class declaration

class ProgressMainWindow: public VkSimpleWindow {

public:

    ProgressMainWindow( const char * name,
                        ArgList args = NULL,
                        Cardinal argCount = 0 );
    ~ProgressMainWindow();
    const char *className();
    virtual Boolean okToQuit();

    /*--- Start editable code block: ProgressMainWindow public

    Boolean get_status() {return ((ProgressBB *)_progressBB) -> _cancel;}

    void init(char *msg) {((ProgressBB *)_progressBB) -> init(msg);}
    void set_title(char *msg) {((ProgressBB *)_progressBB) -> set_title(msg);}
    void update_percent(int c, int n) {((ProgressBB *)_progressBB) -> update_percent(c,

    /*--- End editable code block: ProgressMainWindow public

protected:

    // Classes created by this class

    class ProgressBB *_progressBB;

    // Widgets created by this class

    /*--- Start editable code block: ProgressMainWindow protected

```

```
//---- End editable code block: ProgressMainWindow protected
```

```
private:
```

```
static String _defaultProgressMainWindowResources[];
```

```
//---- Start editable code block: ProgressMainWindow private
```

```
//---- End editable code block: ProgressMainWindow private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```
#ifndef UTILITY_3D_H
#define UTILITY_3D_H
```

646

```
#include <Vk/VkComponent.h>
```

```
#include <Inventor/Xt/viewers/SoXtExaminerViewer.h>
```

```
#include <Inventor/So.h> // Includes ALL Inventor headers
```

```
// Replace for efficiency and faster compilation
```

```
#include <Inventor/Xt/SoXt.h>
```

```
#include <Inventor/Xt/SoXtRenderArea.h>
```

```
#include <Inventor/nodes/SoSeparator.h>
```

```
#include "Points.h"
```

```
class Utility_3D
```

```
{
```

```
public:
```

```
Utility_3D ();
```

```
~Utility_3D();
```

```
class SoXtExaminerViewer *create_iv(char *name, Widget wid, class SoXtExaminerViewe  
int x, int y, int w, int h);
```

```
class SoXtRenderArea *create_localizer_iv(char *name, Widget wid, class SoXtRender?  
int x, int y, int w, int h, SoSeparator *root);
```

```
void to_ivFile(int w,int h,short **img, float *ratio, int, float *, float *, int fi
```

```
void to_ivFileAnimate(int w,int h,short **img, float *ratio, int, float *, int flow
```

```
void to_ivFileAnimateRot(int frame, int w,int h,short **img, float *ratio, int, flc
```

```
void to_ivFile(int w,int h,short **img, Points *, int flow_dir=1);
```

```
private:
```

```
};
```

```
#endif
```

```
#ifndef CYLINDER_H
#define CYLINDER_H
```

647

```
#include <Vk/VkComponent.h>
#include <stdio.h>
#include "Points.h"
```

```
class Cylinder
{
```

```
    public:
```

```
        Cylinder ();
        ~Cylinder();
```

```
        void add(int index_z, Points *p);
```

```
        void clear();
```

```
        void uniform_Points();
```

```
        void to_ivFileContour(FILE *fp, float thickness, float pX, float pY, int index_obj);
```

```
        void to_ivFileSurface(FILE *fp, float thickness, float pX, float pY, int index_obj);
```

```
        void oneLayer(FILE *fp, int i, int num);
```

```
        int _numFrames;
```

```
        int _numPoints;
```

```
        Points _plane[400];
```

```
        int _z[400];
```

```
    private:
```

```
};
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for Win3DMainWindow
//
// This class is a subclass of VkSimpleWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
////////////////////////////////////
#ifndef WIN3DMAINWINDOW_H
#define WIN3DMAINWINDOW_H
#include <Vk/VkSimpleWindow.h>

//--- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Inventor/Sb.h>
#include <Inventor/nodes/SoSeparator.h>
#include <Inventor/nodes/SoTransform.h>

#define _LOC_PLANE 0
#define _LOC_VESSELS 1
#define _LOC_UNIVERSE 2

int _FaceIndex[5] = {0, 1, 2, 3, -1};

//--- End editable code block: headers and declarations

//--- Win3DMainWindow class declaration

class Win3DMainWindow: public VkSimpleWindow {
public:
    Win3DMainWindow( const char * name,
                    ArgList args = NULL,
                    Cardinal argCount = 0 );
    ~Win3DMainWindow();
    const char *className();
    virtual Boolean okToQuit();

    //--- Start editable code block: Win3DMainWindow public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void update();
    void update_localizer(SoSeparator *root);

    void set(int which) {_whichScene = which;}
    int get_whichScene() {return _whichScene;}

    float _planeVertex[4][3];

    void update_plane();

```

```

SoSeparator *get_root() {return _root;}

void clear();

//---- End editable code block: Win3DMainWindow public

protected:

// Classes created by this class
class SoXtRenderArea *_viewer;

// Widgets created by this class

//---- Start editable code block: Win3DMainWindow protected
virtual SbBool appEventHandler(void *userData, XAnyEvent *anyevent);
//---- End editable code block: Win3DMainWindow protected

private:

static String _defaultWin3DMainWindowResources[];

//---- Start editable code block: Win3DMainWindow private
int      _whichScene;
int      _x1, _y1, _x2, _y2;
Boolean  _button1, _button2, _button3;
SoSeparator *_root;

void init();

void mouse(int, int, int, int, int);
void act_transform(int whichMouse, SoTransform *myTransform,
                  float x, float y, float d);

static SbBool myAppEventHandler(void *userData, XAnyEvent *anyevent);

//---- End editable code block: Win3DMainWindow private

};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```

User: meide
Host: phoenix
Class: phoenix
Job: ProgressMainWindow.h

650

```

////////////////////////////////////
//
// Header file for Bb
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BB_H
#define BB_H
#include "BbUI.h"
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- Bb class declaration

class Bb : public BbUI
{
public:
    Bb ( const char *, Widget );
    Bb ( const char * );
    ~Bb();
    const char * className();
    virtual void copy();
    virtual void cut();
    virtual void expertCallback(Widget, XtPointer);
    virtual void imgInfoCallback(Widget, XtPointer);
    virtual void mraInfoCallback(Widget, XtPointer);
    virtual void newFile();
    virtual void noviesCallback(Widget, XtPointer);
    virtual void openFile(const char *);
    virtual void paste();
    virtual void pcmraCutCallback(Widget, XtPointer);
    virtual void print(const char *);
    virtual void save();
    virtual void saveas(const char *);

    static VkComponent *CreateBb( const char *name, Widget parent );

//---- Start editable code block: Bb public

//---- End editable code block: Bb public

protected:

```



```
// These functions will be called as a result of callbacks
// registered in BbUI
```

```
virtual void Next ( Widget, XtPointer );
virtual void Prev ( Widget, XtPointer );
virtual void doOption3D ( Widget, XtPointer );
virtual void doOptionAnimate ( Widget, XtPointer );
virtual void doOptionColor2D ( Widget, XtPointer );
virtual void doOptionGray2D ( Widget, XtPointer );
virtual void doOptionMagnitude ( Widget, XtPointer );
virtual void doOptionNewAnimate ( Widget, XtPointer );
virtual void doOptionOther ( Widget, XtPointer );
virtual void doOptionPhase ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionReference ( Widget, XtPointer );
virtual void doOptionSimple ( Widget, XtPointer );
virtual void doOptionSpline ( Widget, XtPointer );
virtual void doOptionStopAnimate ( Widget, XtPointer );
virtual void doOptionVelocity ( Widget, XtPointer );
virtual void doOptionWhole ( Widget, XtPointer );
```

```
//---- Start editable code block: Bb protected
```

```
//---- End editable code block: Bb protected
```

```
private:
```

```
static void* RegisterBbInterface();
```

```
//---- Start editable code block: Bb private
```

```
//---- End editable code block: Bb private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for Bb3D
//
// This file is generated by RapidApp 1.2
//
// This class is derived from Bb3DUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BB3D_H
#define BB3D_H
#include "Bb3DUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- Bb3D class declaration

class Bb3D : public Bb3DUI
{
public:

    Bb3D ( const char *, Widget );
    Bb3D ( const char * );
    ~Bb3D();
    const char * className();

    static VkComponent *CreateBb3D( const char *name, Widget parent );

    //---- Start editable code block: Bb3D public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void update();

    //---- End editable code block: Bb3D public

protected:

    // These functions will be called as a result of callbacks
    // registered in Bb3DUI

    virtual void HeightDn ( Widget, XtPointer );
    virtual void HeightUp ( Widget, XtPointer );

```

```

virtual void High3D ( Widget, XtPointer );
virtual void Low3D ( Widget, XtPointer );
virtual void TextHeight3D ( Widget, XtPointer );
virtual void TextYPos ( Widget, XtPointer );
virtual void YPosDn ( Widget, XtPointer );
virtual void YPosUp ( Widget, XtPointer );
virtual void doButtonNormalize ( Widget, XtPointer );
virtual void doOptionFixed ( Widget, XtPointer );
virtual void doOptionFlowASIS ( Widget, XtPointer );
virtual void doOptionFlowReverse ( Widget, XtPointer );
virtual void doOptionOrthoCamera ( Widget, XtPointer );
virtual void doOptionPersCamera ( Widget, XtPointer );
virtual void doOptionPersCameraRot ( Widget, XtPointer );
virtual void doOptionSetting3D ( Widget, XtPointer );

```

```
//---- Start editable code block: Bb3D protected
```

```
//---- End editable code block: Bb3D protected
```

```
private:
```

```
static void* RegisterBb3DInterface();
```

```
//---- Start editable code block: Bb3D private
```

```
//---- End editable code block: Bb3D private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for Bb3DUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, Bb3D is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the Bb3D files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BB3DUI_H
#define BB3DUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class Bb3DUI : public VkComponent
{
public:

    Bb3DUI ( const char *, Widget );
    Bb3DUI ( const char * );
    ~Bb3DUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: Bb3D public

    //---- End editable code block: Bb3D public

protected:

    // Widgets created by this class

    Widget _arrowHeightDn;
    Widget _arrowHeightUp;
    Widget _arrowYPosDn;
    Widget _arrowYPosUp;

```

```

Widget  _bb3D;
Widget  _buttonNormal;
Widget  _labelHigh;
Widget  _labelHigh3D;
Widget  _labelLow;
Widget  _labelLow3D;
Widget  _textFieldHeight3D;
Widget  _textFieldHeightFactor;
Widget  _textFieldHigh3D;
Widget  _textFieldLow3D;
Widget  _textFieldYPos;

```

```

VkOptionMenu  *_optionMenu;
VkOptionMenu  *_optionMenu1;
VkOptionMenu  *_optionMenu2;

```

```

VkMenuItem  *_optionFixed;
VkMenuItem  *_optionFlowASIS;
VkMenuItem  *_optionFlowReverse;
VkMenuItem  *_optionOrthoCamera;
VkMenuItem  *_optionPersCamera;
VkMenuItem  *_optionPersCameraRot;
VkMenuItem  *_optionSetting3D;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void HeightDn ( Widget, XtPointer );
virtual void HeightUp ( Widget, XtPointer );
virtual void High3D ( Widget, XtPointer );
virtual void Low3D ( Widget, XtPointer );
virtual void TextHeight3D ( Widget, XtPointer );
virtual void TextYPos ( Widget, XtPointer );
virtual void YPosDn ( Widget, XtPointer );
virtual void YPosUp ( Widget, XtPointer );
virtual void doButtonNormalize ( Widget, XtPointer );
virtual void doOptionFixed ( Widget, XtPointer );
virtual void doOptionFlowASIS ( Widget, XtPointer );
virtual void doOptionFlowReverse ( Widget, XtPointer );
virtual void doOptionOrthoCamera ( Widget, XtPointer );
virtual void doOptionPersCamera ( Widget, XtPointer );
virtual void doOptionPersCameraRot ( Widget, XtPointer );
virtual void doOptionSetting3D ( Widget, XtPointer );

```

```

//---- Start editable code block: Bb3D protected

```

```

//---- End editable code block: Bb3D protected

```

```

private:

```

```

// Array of default resources

```

```

static String  _defaultBb3DUIResources[];

```

```

// Callbacks to interface with Motif

```

```

static void HeightDnCallback ( Widget, XtPointer, XtPointer );
static void HeightUpCallback ( Widget, XtPointer, XtPointer );
static void High3DCallback ( Widget, XtPointer, XtPointer );
static void Low3DCallback ( Widget, XtPointer, XtPointer );
static void TextHeight3DCallback ( Widget, XtPointer, XtPointer );
static void TextYPosCallback ( Widget, XtPointer, XtPointer );

```

```
static void YPosDnCallback ( Widget, XtPointer, XtPointer );
static void YPosUpCallback ( Widget, XtPointer, XtPointer );
static void doButtonNormalizeCallback ( Widget, XtPointer, XtPointer );
static void doOptionFixedCallback ( Widget, XtPointer, XtPointer );
static void doOptionFlowASISCallback ( Widget, XtPointer, XtPointer );
static void doOptionFlowReverseCallback ( Widget, XtPointer, XtPointer );
static void doOptionOrthoCameraCallback ( Widget, XtPointer, XtPointer );
static void doOptionPersCameraCallback ( Widget, XtPointer, XtPointer );
static void doOptionPersCameraRotCallback ( Widget, XtPointer, XtPointer );
static void doOptionSetting3DCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: Bb3D private
```

```
//---- End editable code block: Bb3D private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbAnimation
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbAnimationUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBANIMATION_H
#define BBANIMATION_H
#include "BbAnimationUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbAnimation class declaration
class BbAnimation : public BbAnimationUI
{
public:
    BbAnimation ( const char *, Widget );
    BbAnimation ( const char * );
    ~BbAnimation();
    const char * className();

    static VkComponent *CreateBbAnimation( const char *name, Widget parent );

//---- Start editable code block: BbAnimation public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    Widget get_textfield() {return _textfield;}

    void init();
    void set_toggle(int);

//---- End editable code block: BbAnimation public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbAnimationUI

```

```
virtual void animate1 ( Widget, XtPointer );
virtual void doButtonHeart ( Widget, XtPointer );
virtual void setToggle1D ( Widget, XtPointer );
virtual void setToggle2D ( Widget, XtPointer );
virtual void setToggle3D ( Widget, XtPointer );
virtual void setToggleSymphony ( Widget, XtPointer );
```

```
//---- Start editable code block: BbAnimation protected
```

```
//---- End editable code block: BbAnimation protected
```

```
private:
```

```
static void* RegisterBbAnimationInterface();
```

```
//---- Start editable code block: BbAnimation private
```

```
//---- End editable code block: BbAnimation private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```



```

////////////////////////////////////
//
// Header file for BbAnimationUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbAnimation is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbAnimation files
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBANIMATIONUI_H
#define BBANIMATIONUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class BbAnimationUI : public VkComponent
{
public:
    BbAnimationUI ( const char *, Widget );
    BbAnimationUI ( const char * );
    ~BbAnimationUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbAnimation public

    //---- End editable code block: BbAnimation public

protected:

    // Widgets created by this class

    Widget _bbAnimation;
    Widget _buttonHeart;
    Widget _labelTime;
    Widget _radioboxAnimate;
    Widget _textfield;
    Widget _toggle2D;
    Widget _toggle3D;
    Widget _toggleFlow;

```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

virtual void animateTime ( Widget, XtPointer );
virtual void doButtonHeart ( Widget, XtPointer );
virtual void setToggle1D ( Widget, XtPointer );
virtual void setToggle2D ( Widget, XtPointer );
virtual void setToggle3D ( Widget, XtPointer );
virtual void setToggleSymphony ( Widget, XtPointer );

//----- Start editable code block: BbAnimation protected

//----- End editable code block: BbAnimation protected

private:

// Array of default resources

static String      _defaultBbAnimationUIResources[];

// Callbacks to interface with Motif

static void animateTimeCallback ( Widget, XtPointer, XtPointer );
static void doButtonHeartCallback ( Widget, XtPointer, XtPointer );
static void setToggle1DCallback ( Widget, XtPointer, XtPointer );
static void setToggle2DCallback ( Widget, XtPointer, XtPointer );
static void setToggle3DCallback ( Widget, XtPointer, XtPointer );
static void setToggleSymphonyCallback ( Widget, XtPointer, XtPointer );

//----- Start editable code block: BbAnimation private

//----- End editable code block: BbAnimation private
};
//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

#endif
```

```

////////////////////////////////////
//
// Header file for BbDetail
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbDetailUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBDETAIL_H
#define BBDETAIL_H
#include "BbDetailUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbDetail class declaration

class BbDetail : public BbDetailUI
{
public:
    BbDetail ( const char *, Widget );
    BbDetail ( const char * );
    ~BbDetail();
    const char * className();

    static VkComponent *CreateBbDetail( const char *name, Widget parent );

//---- Start editable code block: BbDetail public
    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void init();

    void set(int, int, int, int, int, int);

//---- End editable code block: BbDetail public

protected:

    // These functions will be called as a result of callbacks

```

```
// registered in BbDetailUI

virtual void setToggleHide ( Widget, XtPointer );
virtual void setToggleShow ( Widget, XtPointer );

//---- Start editable code block: BbDetail protected

//---- End editable code block: BbDetail protected

private:

    static void* RegisterBbDetailInterface();

    //---- Start editable code block: BbDetail private

    //---- End editable code block: BbDetail private

};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif
```

```

////////////////////////////////////
//
// Header file for BbDetailUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbDetail is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbDetail files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBDETAILUI_H
#define BBDETAILUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class BbDetailUI : public VkComponent
{
public:
    BbDetailUI ( const char *, Widget );
    BbDetailUI ( const char * );
    ~BbDetailUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbDetail public

    //---- End editable code block: BbDetail public

protected:

    // Widgets created by this class

    Widget _bbDetail;
    Widget _labelB;
    Widget _labelG;
    Widget _labelR;
    Widget _labelSignal;
    Widget _labelX;
    Widget _labelY;
    Widget _labelZ;

```

```

Widget  _radioboxPixel;
Widget  _textfieldB;
Widget  _textfieldG;
Widget  _textfieldR;
Widget  _textfieldSignal;
Widget  _textfieldX;
Widget  _textfieldY;
Widget  _textfieldZ;
Widget  _toggleHide;
Widget  _toggleShow;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void setToggleHide ( Widget, XtPointer );
virtual void setToggleShow ( Widget, XtPointer );

```

```

//---- Start editable code block: BbDetail protected

```

```

//---- End editable code block: BbDetail protected

```

```

private:

```

```

// Array of default resources

```

```

static String      _defaultBbDetailUIResources[];

```

```

// Callbacks to interface with Motif

```

```

static void setToggleHideCallback ( Widget, XtPointer, XtPointer );
static void setToggleShowCallback ( Widget, XtPointer, XtPointer );

```

```

//---- Start editable code block: BbDetail private

```

```

//---- End editable code block: BbDetail private

```

```

};
//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

```

#endif

```

```

////////////////////////////////////
//
// Header file for BbDisplay
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbDisplayUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBDISPLAY_H
#define BBDISPLAY_H
#include "BbDisplayUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbDisplay class declaration

class BbDisplay : public BbDisplayUI
{
public:
    BbDisplay ( const char *, Widget );
    BbDisplay ( const char * );
    ~BbDisplay();
    const char * className();

    static VkComponent *CreateBbDisplay( const char *name, Widget parent );

//---- Start editable code block: BbDisplay public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void init();
//---- End editable code block: BbDisplay public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbDisplayUI

    virtual void doOptionDouble ( Widget, XtPointer );
    virtual void doOptionFlow2D ( Widget, XtPointer );
    virtual void doOptionFlow3D ( Widget, XtPointer );
    virtual void doOptionHalf ( Widget, XtPointer );

```

```

virtual void doOption2D ( Widget, XtPointer );
virtual void doOptionNormal ( Widget, XtPointer );
virtual void doOptionWhole2D ( Widget, XtPointer );
virtual void imgNum ( Widget, XtPointer );
virtual void imgZoom ( Widget, XtPointer );
virtual void setToggleBoth ( Widget, XtPointer );
virtual void setToggleCombo ( Widget, XtPointer );
virtual void setToggleLeft ( Widget, XtPointer );
virtual void setToggleNormal ( Widget, XtPointer );
virtual void setToggleRight ( Widget, XtPointer );

//---- Start editable code block: BbDisplay protected

//---- End editable code block: BbDisplay protected

private:

static void* RegisterBbDisplayInterface();

//---- Start editable code block: BbDisplay private

//---- End editable code block: BbDisplay private

};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```



```

////////////////////////////////////
//
// Header file for BbDisplayUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbDisplay is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbDisplay files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBDISPLAYUI_H
#define BBDISPLAYUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class BbDisplayUI : public VkComponent
{
public:

    BbDisplayUI ( const char *, Widget );
    BbDisplayUI ( const char * );
    ~BbDisplayUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbDisplay public

    Widget _bbDisplay;
    Widget _labelDisplayTotal;
    Widget _labelDisplayTotalNum;
    Widget _labelImageNumber;
    Widget _labelZoom;
    Widget _radiobox4;
    Widget _radioboxZoom;
    Widget _separator2;
    Widget _textfieldDisplayImgNumber;
    Widget _textfieldZoom;
    Widget _toggleBoth;
    Widget _toggleCombo;
    Widget _toggleLeft;

```

```
Widget _toggleNormal;  
Widget _toggleRight;
```

669

```
//----- End editable code block: BbDisplay public
```

protected:

```
// Widgets created by this class
```

```
VkOptionMenu *_optionMenu3;  
VkOptionMenu *_optionMenuZoom;
```

```
VkMenuItem *_optionDouble;  
VkMenuItem *_optionFlow2D;  
VkMenuItem *_optionFlow3D;  
VkMenuItem *_optionHalf;  
VkMenuItem *_optionL3D;  
VkMenuItem *_optionNormal;  
VkMenuItem *_optionWhole2D;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doOptionDouble ( Widget, XtPointer );  
virtual void doOptionFlow2D ( Widget, XtPointer );  
virtual void doOptionFlow3D ( Widget, XtPointer );  
virtual void doOptionHalf ( Widget, XtPointer );  
virtual void doOptionL3D ( Widget, XtPointer );  
virtual void doOptionNormal ( Widget, XtPointer );  
virtual void doOptionWhole2D ( Widget, XtPointer );  
virtual void imgNum ( Widget, XtPointer );  
virtual void imgZoom ( Widget, XtPointer );  
virtual void setToggleBoth ( Widget, XtPointer );  
virtual void setToggleCombo ( Widget, XtPointer );  
virtual void setToggleLeft ( Widget, XtPointer );  
virtual void setToggleNormal ( Widget, XtPointer );  
virtual void setToggleRight ( Widget, XtPointer );
```

```
//----- Start editable code block: BbDisplay protected
```

```
//----- End editable code block: BbDisplay protected
```

private:

```
// Array of default resources
```

```
static String _defaultBbDisplayUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doOptionDoubleCallback ( Widget, XtPointer, XtPointer );  
static void doOptionFlow2DCallback ( Widget, XtPointer, XtPointer );  
static void doOptionFlow3DCallback ( Widget, XtPointer, XtPointer );  
static void doOptionHalfCallback ( Widget, XtPointer, XtPointer );  
static void doOptionL3DCallback ( Widget, XtPointer, XtPointer );  
static void doOptionNormalCallback ( Widget, XtPointer, XtPointer );  
static void doOptionWhole2DCallback ( Widget, XtPointer, XtPointer );
```

```
static void imgNumCallback ( Widget, XtPointer, XtPointer );
static void imgZoomCallback ( Widget, XtPointer, XtPointer );
static void setToggleBtnCallback ( Widget, XtPointer, XtPointer );
static void setToggleComboCallback ( Widget, XtPointer, XtPointer );
static void setToggleLeftCallback ( Widget, XtPointer, XtPointer );
static void setToggleNormalCallback ( Widget, XtPointer, XtPointer );
static void setToggleRightCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbDisplay private
```

```
//---- End editable code block: BbDisplay private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbFlow
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFlowUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBFLOW_H
#define BBFLOW_H
#include "BbFlowUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbFlow class declaration

class BbFlow : public BbFlowUI
{
public:
    BbFlow ( const char *, Widget );
    BbFlow ( const char * );
    ~BbFlow();
    const char * className();

    static VkComponent *CreateBbFlow( const char *name, Widget parent );

//---- Start editable code block: BbFlow public
    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void set_noiseLevel(int noise, float noiseflow);

//---- End editable code block: BbFlow public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbFlowUI

    virtual void SemiFlow ( Widget, XtPointer );
    virtual void SemiFlowChg ( Widget, XtPointer );

```

```
virtual void doButtonAcceptFlow ( Widget, XtPointer );
virtual void doOptionAcceptEdge ( Widget, XtPointer );
virtual void doOptionAutoSnake ( Widget, XtPointer );
virtual void doOptionAutoThresh ( Widget, XtPointer );
virtual void doOptionManual ( Widget, XtPointer );
virtual void doOptionSemiAuto ( Widget, XtPointer );
virtual void maxFlow ( Widget, XtPointer );
virtual void minFlow ( Widget, XtPointer );
```

```
//---- Start editable code block: BbFlow protected
```

```
//---- End editable code block: BbFlow protected
```

```
private:
```

```
static void* RegisterBbFlowInterface();
```

```
//---- Start editable code block: BbFlow private
```

```
//---- End editable code block: BbFlow private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbFlow3D
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFlow3DUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBFLOW3D_H
#define BBFLOW3D_H
#include "BbFlow3DUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "ImgGE.h"

#define MAXPTS 1000000
#define MAXINDEX 5000000

static int _boxLineIndex[36] = {
    0, 1, -1,
    1, 2, -1,
    2, 3, -1,
    3, 0, -1,
    4, 5, -1,
    5, 6, -1,
    6, 7, -1,
    7, 4, -1,
    3, 7, -1,
    2, 6, -1,
    1, 5, -1,
    0, 4, -1
};

//---- End editable code block: headers and declarations

//---- BbFlow3D class declaration

class BbFlow3D : public BbFlow3DUI
{
public:
    BbFlow3D ( const char *, Widget );
    BbFlow3D ( const char * );
    ~BbFlow3D();
    const char * className();

    static VkComponent *CreateBbFlow3D( const char *name, Widget parent );

//---- Start editable code block: BbFlow3D public

```

```

ObjectManager *_objMag;
void set(ObjectManager *objMag) {_objMag = objMag;}

int  _pts;
int  _indexNum;
int  _index[MAXINDEX];

float _rgb[MAXPTS][3];
float _vertex[MAXPTS][3];

short _data3d[512][512][200];
float _box[8][3];

int getJointPoint_2LinesIn3DSpace(float x1, float y1, float z1, float p1, float q1,
    float x2, float y2, float z2, float p2, float q2, float r2, float *x, float *y, f

int _width;
int _height;
int _depth;
float _pX;
float _pY;
float _thick;
float _tlh_R;
float _tlh_A;
float _tlh_S;

float _planeVertex[4][3];

void get_axialPlaneVertex(int number);
void get_sagitalPlaneVertex(int number);
void get_coronalPlaneVertex(int number);

ImgGE *get_axialPlaneImg(int number);
ImgGE *get_sagitalPlaneImg(int number);
ImgGE *get_coronalPlaneImg(int number);

void get_axial(int number, float x1, float y1,
    float *x, float *y, float *z);
void get_sagital(int number, float x1, float y1,
    float *x, float *y, float *z);
void get_coronal(int number, float x1, float y1,
    float *x, float *y, float *z);
void get_anyCut(float *p1, float *p2, float *p3, float *q1,
    float *q2, float *q3, float *r1, float *r2, float *r3);
void get_transform(float x1, float y1, float *x2, float *y2,
    float *x, float *y, float *z);

void get_axialCenter(int number, unsigned char **area,
    float *x, float *y, float *z);
void get_sagitalCenter(int number, unsigned char **area,
    float *x, float *y, float *z);
void get_coronalCenter(int number, unsigned char **area,
    float *x, float *y, float *z);

void addCut();

//---- End editable code block: BbFlow3D public

protected:

// These functions will be called as a result of callbacks

```

```

virtual void doButtonAccept ( Widget, XtPointer );
virtual void doButtonSaveFlow3D ( Widget, XtPointer );
virtual void doOptionAddCut ( Widget, XtPointer );
virtual void doOptionAddObj ( Widget, XtPointer );
virtual void doOptionPoint ( Widget, XtPointer );
virtual void doOptionSurface ( Widget, XtPointer );
virtual void setToggleDisable ( Widget, XtPointer );
virtual void setToggleEnable ( Widget, XtPointer );

```

```
//---- Start editable code block: BbFlow3D protected
```

```
//---- End editable code block: BbFlow3D protected
```

```
private:
```

```

static void* RegisterBbFlow3DInterface();

//---- Start editable code block: BbFlow3D private

float  _lowThreshold, _highThreshold;
float  _xc, _yc, _zc;
float  _low, _xlow, _ylow, _zlow;
float  _high, _xhigh, _yhigh, _zhigh;

int     p[8][3], connect[8][3], flag[8];
float   _isoThreshold;
float   res[6][3];

void pointMaker(Widget);
void pointMaker0();
void sceneMaker(Widget);

void surfaceMaker(Widget);
void surfaceMaker0();

Boolean threshold(short data);
Boolean borderPoint(int x, int y, int z);
Boolean interPoint(int x, int y, int z);

void add_to_scene(int k);

void add_res(float *mid, int k);

void marchingCube();
void marchingCube1(int im);
void marchingCube2(int im1, int im2);
void marchingCube3(int im1, int im2, int im3);
void marchingCube44(int im1, int im2, int im3, int im4);
void marchingCube430(int im1, int im2, int im3, int im4);
void marchingCube431(int im1, int im2, int im3, int im4);

void interpolatePoint(int im1, int im2, float *mid);
Boolean neighbor2(int i1, int i2);
int neighbor3(int *im1, int *im2, int *im3);
int neighbor4(int *im1, int *im2, int *im3, int *im4);

void disp_info(int n);
void add_coord(int n);
void set_scene();

```


//---- End editable code block: BbFlow3D private

676

```
};  
//---- Start editable code block: End of generated code  
  
//---- End editable code block: End of generated code  
#endif
```

```

/////////////////////////////////////////////////////////////////
//
// Header file for BbFlow3DUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbFlow3D is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbFlow3D files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
/////////////////////////////////////////////////////////////////
#ifndef BBFLOW3DUI_H
#define BBFLOW3DUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class VkMenuItem;
class BbFlow3DUI : public VkComponent
{
public:
    BbFlow3DUI ( const char *, Widget );
    BbFlow3DUI ( const char * );
    ~BbFlow3DUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbFlow3D public

    //---- End editable code block: BbFlow3D public

protected:

    // Widgets created by this class

    Widget _bbFlow3D;
    Widget _buttonAccept1;
    Widget _buttonSaveFlow3D;
    Widget _labelFlow3DEnd;

```

```

Widget _labelFlow3DHigh;
Widget _labelFlow3DLow;
Widget _labelFlow3DStart;
Widget _radiobox;
Widget _textfield1;
Widget _textfieldFlow3DEnd;
Widget _textfieldFlow3DHigh;
Widget _textfieldFlow3DLow;
Widget _textfieldFlow3DStart;
Widget _toggleDisable;
Widget _toggleEnable;

```

```

VkOptionMenu *_optionMenu10;
VkOptionMenu *_optionMenu12;

```

```

VkMenuItem *_optionAddCut;
VkMenuItem *_optionAddObj;
VkMenuItem *_optionPoint;
VkMenuItem *_optionSurface;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void doButtonAccept ( Widget, XtPointer );
virtual void doButtonSaveFlow3D ( Widget, XtPointer );
virtual void doOptionAddCut ( Widget, XtPointer );
virtual void doOptionAddObj ( Widget, XtPointer );
virtual void doOptionPoint ( Widget, XtPointer );
virtual void doOptionSurface ( Widget, XtPointer );
virtual void setToggleDisable ( Widget, XtPointer );
virtual void setToggleEnable ( Widget, XtPointer );

```

```

//---- Start editable code block: BbFlow3D protected

```

```

//---- End editable code block: BbFlow3D protected

```

```

private:

```

```

// Array of default resources

```

```

static String _defaultBbFlow3DUIResources[];

```

```

// Callbacks to interface with Motif

```

```

static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );
static void doButtonSaveFlow3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionAddCutCallback ( Widget, XtPointer, XtPointer );
static void doOptionAddObjCallback ( Widget, XtPointer, XtPointer );
static void doOptionPointCallback ( Widget, XtPointer, XtPointer );
static void doOptionSurfaceCallback ( Widget, XtPointer, XtPointer );
static void setToggleDisableCallback ( Widget, XtPointer, XtPointer );
static void setToggleEnableCallback ( Widget, XtPointer, XtPointer );

```

```

//---- Start editable code block: BbFlow3D private

```

```

//---- End editable code block: BbFlow3D private

```

```

};
//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

#endif

679

User: meide
Host: phoenix
Class: phoenix
Job: BbDisplay.h

680

```

////////////////////////////////////
//
// Header file for BbFlowUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbFlow is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbFlow files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBFLOWUI_H
#define BBFLOWUI_H
#include <Vk/VkComponent.h>

//----- Start editable code block: headers and declarations

//----- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbFlowUI : public VkComponent
{
public:

    BbFlowUI ( const char *, Widget );
    BbFlowUI ( const char * );
    ~BbFlowUI();
    void create ( Widget );
    const char * className();

    //----- Start editable code block: BbFlow public

    Widget _bbFlow;
    Widget _buttonAcceptFlow;
    Widget _labelArea;
    Widget _labelBSV;
    Widget _labelMV;
    Widget _labelPSV;
    Widget _labelVFR;
    Widget _textfieldArea;
    Widget _textfieldBSV;
    Widget _textfieldMV;
    Widget _textfieldMax;
    Widget _textfieldMin;
    Widget _textfieldPSV;

```

//---- End editable code block: BbFlow public

protected:

// Widgets created by this class

VkOptionMenu *_optionMenuFlowMethod;

VkMenuItem *_optionAutoEdge;
VkMenuItem *_optionAutoSnake;
VkMenuItem *_optionAutoThresh;
VkMenuItem *_optionManual;
VkMenuItem *_optionSemiAuto;

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

virtual void SemiFlow (Widget, XtPointer);
virtual void SemiFlowChg (Widget, XtPointer);
virtual void doButtonAcceptFlow (Widget, XtPointer);
virtual void doOptionAutoEdge (Widget, XtPointer);
virtual void doOptionAutoSnake (Widget, XtPointer);
virtual void doOptionAutoThresh (Widget, XtPointer);
virtual void doOptionManual (Widget, XtPointer);
virtual void doOptionSemiAuto (Widget, XtPointer);
virtual void maxFlow (Widget, XtPointer);
virtual void minFlow (Widget, XtPointer);

//---- Start editable code block: BbFlow protected

//---- End editable code block: BbFlow protected

private:

// Array of default resources

static String _defaultBbFlowUIResources[];

// Callbacks to interface with Motif

static void SemiFlowCallback (Widget, XtPointer, XtPointer);
static void SemiFlowChgCallback (Widget, XtPointer, XtPointer);
static void doButtonAcceptFlowCallback (Widget, XtPointer, XtPointer);
static void doOptionAutoEdgeCallback (Widget, XtPointer, XtPointer);
static void doOptionAutoSnakeCallback (Widget, XtPointer, XtPointer);
static void doOptionAutoThreshCallback (Widget, XtPointer, XtPointer);
static void doOptionManualCallback (Widget, XtPointer, XtPointer);
static void doOptionSemiAutoCallback (Widget, XtPointer, XtPointer);
static void maxFlowCallback (Widget, XtPointer, XtPointer);
static void minFlowCallback (Widget, XtPointer, XtPointer);

//---- Start editable code block: BbFlow private

//---- End editable code block: BbFlow private

```
};  
//---- Start editable code block: End of generated code
```

683

```
//---- End editable code block: End of generated code  
#endif
```



```

////////////////////////////////////
//
// Header file for BbFormat
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbFormatUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBFORMAT_H
#define BBFORMAT_H
#include "BbFormatUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "Utility_Vision.h"

#define IMAGE_WHOLE 0
#define IMAGE_ROI 1
#define IMAGE_3DLOCLARGE 2
#define IMAGE_3DLOCSMALL 3
#define IMAGE_3DFLOWLARGE 4
#define IMAGE_3DFLOWSMALL 5

#define IMAGE_BIN 0
#define IMAGE_RGB 1
#define IMAGE_GIF 2
#define IMAGE_TIFF 3

//---- End editable code block: headers and declarations

//---- BbFormat class declaration

class BbFormat : public BbFormatUI
{
public:
    BbFormat ( const char *, Widget );
    BbFormat ( const char * );
    ~BbFormat();
    const char * className();

    static VkComponent *CreateBbFormat( const char *name, Widget parent );

//---- Start editable code block: BbFormat public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    float _avgFlow;
    float _radius;

```

```
float _shear;
```

```
void toFile(char *fname, int w, int h, XImage *ximage);
void toHTMLFile(char *fname, int type);
void toMPEG();
float toFlowFile();
void updatePCMR_HTML(float avg);
```

```
void savePublish(int);
void setPath(char *path);
```

```
//---- End editable code block: BbFormat public
```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbFormatUI
```

```
virtual void doButton3DContour ( Widget, XtPointer );
virtual void doButtonAcceptFlow ( Widget, XtPointer );
virtual void doButtonPublishPath ( Widget, XtPointer );
virtual void doButtonShow2DContour ( Widget, XtPointer );
virtual void doOption2DLoc ( Widget, XtPointer );
virtual void doOption2DMag ( Widget, XtPointer );
virtual void doOption2DPhase ( Widget, XtPointer );
virtual void doOption2DWave ( Widget, XtPointer );
virtual void doOption3DFlow ( Widget, XtPointer );
virtual void doOption3DFlowLarge ( Widget, XtPointer );
virtual void doOption3DFlowSmall ( Widget, XtPointer );
virtual void doOption3DLoc ( Widget, XtPointer );
virtual void doOption3DLocLarge ( Widget, XtPointer );
virtual void doOption3DLocSmall ( Widget, XtPointer );
virtual void doOptionGIF ( Widget, XtPointer );
virtual void doOptionHTML ( Widget, XtPointer );
virtual void doOptionMPEG ( Widget, XtPointer );
virtual void doOptionPublishArea ( Widget, XtPointer );
virtual void doOptionPublishNone ( Widget, XtPointer );
virtual void doOptionPublishShear ( Widget, XtPointer );
virtual void doOptionRGB ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionTIFF ( Widget, XtPointer );
virtual void doOptionWholeImg ( Widget, XtPointer );
virtual void newPath ( Widget, XtPointer );
```

```
//---- Start editable code block: BbFormat protected
```

```
//---- End editable code block: BbFormat protected
```

private:

```
static void* RegisterBbFormatInterface();
```

```
//---- Start editable code block: BbFormat private
```

```
int _format;
int _type;
```

```
void toFile(char *fname);
void init();
```

//----- End editable code block: BbFormat private

686

```
};  
//----- Start editable code block: End of generated code  
  
//----- End editable code block: End of generated code  
  
#endif
```

```

////////////////////////////////////
//
// Header file for BbFormatUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbFormat is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbFormat files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
-// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBFORMATUI_H
#define BBFORMATUI_H
#include <Vk/VkComponent.h>

//----- Start editable code block: headers and declarations

//----- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbFormatUI : public VkComponent
{
public:
    BbFormatUI ( const char *, Widget );
    BbFormatUI ( const char * );
    ~BbFormatUI();
    void create ( Widget );
    const char * className();

    //----- Start editable code block: BbFormat public

    //----- End editable code block: BbFormat public

protected:

    // Widgets created by this class
    Widget _bbFormat;
    Widget _button3DContour;
    Widget _buttonAcceptFlow1;
    Widget _buttonPublishPath;

```

```
Widget _buttonShow2DContour;
Widget _labelFname;
Widget _labelFnameExt;
Widget _textfieldExtension;
Widget _textfieldFname;
Widget _textfieldNewPath1;
```

```
VkOptionMenu *_optionMenu13;
VkOptionMenu *_optionMenu16;
VkOptionMenu *_optionMenu9;
```

```
VkMenuItem *_option2DLoc;
VkMenuItem *_option2DMag;
VkMenuItem *_option2DPhase;
VkMenuItem *_option2DWave;
VkMenuItem *_option3DFlow;
VkMenuItem *_option3DFlowLarge;
VkMenuItem *_option3DFlowSmall;
VkMenuItem *_option3DLoc;
VkMenuItem *_option3DLocLarge;
VkMenuItem *_option3DLocSmall;
VkMenuItem *_optionGIF;
VkMenuItem *_optionHTML;
VkMenuItem *_optionMPEG;
VkMenuItem *_optionPublishArea;
VkMenuItem *_optionPublishNone;
VkMenuItem *_optionPublishShear;
VkMenuItem *_optionRGB;
VkMenuItem *_optionROI3;
VkMenuItem *_optionTIFF;
VkMenuItem *_optionWholeImg;
```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions
```

```
virtual void doButton3DContour ( Widget, XtPointer );
virtual void doButtonAcceptFlow ( Widget, XtPointer );
virtual void doButtonPublishPath ( Widget, XtPointer );
virtual void doButtonShow2DContour ( Widget, XtPointer );
virtual void doOption2DLoc ( Widget, XtPointer );
virtual void doOption2DMag ( Widget, XtPointer );
virtual void doOption2DPhase ( Widget, XtPointer );
virtual void doOption2DWave ( Widget, XtPointer );
virtual void doOption3DFlow ( Widget, XtPointer );
virtual void doOption3DFlowLarge ( Widget, XtPointer );
virtual void doOption3DFlowSmall ( Widget, XtPointer );
virtual void doOption3DLoc ( Widget, XtPointer );
virtual void doOption3DLocLarge ( Widget, XtPointer );
virtual void doOption3DLocSmall ( Widget, XtPointer );
virtual void doOptionGIF ( Widget, XtPointer );
virtual void doOptionHTML ( Widget, XtPointer );
virtual void doOptionMPEG ( Widget, XtPointer );
virtual void doOptionPublishArea ( Widget, XtPointer );
virtual void doOptionPublishNone ( Widget, XtPointer );
virtual void doOptionPublishShear ( Widget, XtPointer );
virtual void doOptionRGB ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionTIFF ( Widget, XtPointer );
virtual void doOptionWholeImg ( Widget, XtPointer );
virtual void newPath ( Widget, XtPointer );
```

```
//----- Start editable code block: BbFormat protected
```

```
//----- End editable code block: BbFormat protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbFormatUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doButton3DContourCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptFlowCallback ( Widget, XtPointer, XtPointer );
static void doButtonPublishPathCallback ( Widget, XtPointer, XtPointer );
static void doButtonShow2DContourCallback ( Widget, XtPointer, XtPointer );
static void doOption2DLocCallback ( Widget, XtPointer, XtPointer );
static void doOption2DMagCallback ( Widget, XtPointer, XtPointer );
static void doOption2DPhaseCallback ( Widget, XtPointer, XtPointer );
static void doOption2DWaveCallback ( Widget, XtPointer, XtPointer );
static void doOption3DFlowCallback ( Widget, XtPointer, XtPointer );
static void doOption3DFlowLargeCallback ( Widget, XtPointer, XtPointer );
static void doOption3DFlowSmallCallback ( Widget, XtPointer, XtPointer );
static void doOption3DLocCallback ( Widget, XtPointer, XtPointer );
static void doOption3DLocLargeCallback ( Widget, XtPointer, XtPointer );
static void doOption3DLocSmallCallback ( Widget, XtPointer, XtPointer );
static void doOptionGIFCallback ( Widget, XtPointer, XtPointer );
static void doOptionHTMLCallback ( Widget, XtPointer, XtPointer );
static void doOptionMPEGCallback ( Widget, XtPointer, XtPointer );
static void doOptionPublishAreaCallback ( Widget, XtPointer, XtPointer );
static void doOptionPublishNoneCallback ( Widget, XtPointer, XtPointer );
static void doOptionPublishShearCallback ( Widget, XtPointer, XtPointer );
static void doOptionRGBCallback ( Widget, XtPointer, XtPointer );
static void doOptionROICallback ( Widget, XtPointer, XtPointer );
static void doOptionTIFFCallback ( Widget, XtPointer, XtPointer );
static void doOptionWholeImgCallback ( Widget, XtPointer, XtPointer );
static void newPathCallback ( Widget, XtPointer, XtPointer );
```

```
//----- Start editable code block: BbFormat private
```

```
//----- End editable code block: BbFormat private
```

```
};
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```


User: meide
Host: phoenix
Class: phoenix
Job: BbFlowUI.h

691


```

////////////////////////////////////
//
// Header file for BbHistogram
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbHistogramUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBHISTOGRAM_H
#define BBHISTOGRAM_H
#include "BbHistogramUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbHistogram class declaration

class BbHistogram : public BbHistogramUI
{
public:
    BbHistogram ( const char *, Widget );
    BbHistogram ( const char * );
    ~BbHistogram();
    const char * className();

    static VkComponent *CreateBbHistogram( const char *name, Widget parent );

//---- Start editable code block: BbHistogram public

    int _winWidth;
    int _winCenter;

    void init();
    void update(float center, float width);
    void update_width(int);
    void update_center(int);

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

//---- End editable code block: BbHistogram public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in BbHistogramUI
```

```
virtual void abdomen ( Widget, XtPointer );
virtual void bone ( Widget, XtPointer );
virtual void centerDrag ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionLFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung ( Widget, XtPointer );
virtual void mediastinum ( Widget, XtPointer );
virtual void spine ( Widget, XtPointer );
virtual void widthDrag ( Widget, XtPointer );
```

```
//---- Start editable code block: BbHistogram protected
```

```
//---- End editable code block: BbHistogram protected
```

```
private:
```

```
static void* RegisterBbHistogramInterface();
```

```
//---- Start editable code block: BbHistogram private
```

```
//---- End editable code block: BbHistogram private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbHistogramUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbHistogram is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbHistogram files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBHISTOGRAMUI_H
#define BBHISTOGRAMUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuToggle;
class VkMenuItem;
class BbHistogramUI : public VkComponent
{
public:
    BbHistogramUI ( const char *, Widget );
    BbHistogramUI ( const char * );
    ~BbHistogramUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbHistogram public

    Widget _bbHistogram;
    Widget _dialCenter;
    Widget _dialWidth;
    Widget _labelLHistoHigh;
    Widget _labelLHistoLow;
    Widget _labelLHistoMax;
    Widget _labelLHistoMin;

    //--- End editable code block: BbHistogram public

protected:

```

```
// Widgets created by this class
```

```
VkOptionMenu *_optionMenuLHist;
VkOptionMenu *_optionMenuLHistogram;
```

```
VkMenuItem *_optionAbdomen;
VkMenuItem *_optionBone;
VkMenuItem *_optionHead;
VkMenuItem *_optionLCoarse;
VkMenuItem *_optionLFine;
VkMenuItem *_optionLMapping;
VkMenuItem *_optionLUpdate;
VkMenuItem *_optionLung;
VkMenuItem *_optionMediaStinum;
VkMenuItem *_optionSpine;
```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions
```

```
virtual void abdomen ( Widget, XtPointer );
virtual void bone ( Widget, XtPointer );
virtual void centerDrag ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionLFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung ( Widget, XtPointer );
virtual void mediastinum ( Widget, XtPointer );
virtual void spine ( Widget, XtPointer );
virtual void widthDrag ( Widget, XtPointer );
```

```
//----- Start editable code block: BbHistogram protected
```

```
//----- End editable code block: BbHistogram protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbHistogramUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void abdomenCallback ( Widget, XtPointer, XtPointer );
static void boneCallback ( Widget, XtPointer, XtPointer );
static void centerDragCallback ( Widget, XtPointer, XtPointer );
static void doOptionCoarseCallback ( Widget, XtPointer, XtPointer );
static void doOptionLFineCallback ( Widget, XtPointer, XtPointer );
static void doOptionMappingCallback ( Widget, XtPointer, XtPointer );
static void doOptionUpdateCallback ( Widget, XtPointer, XtPointer );
static void headCallback ( Widget, XtPointer, XtPointer );
static void highChgCallback ( Widget, XtPointer, XtPointer );
static void lowChgCallback ( Widget, XtPointer, XtPointer );
static void lungCallback ( Widget, XtPointer, XtPointer );
static void mediastinumCallback ( Widget, XtPointer, XtPointer );
static void spineCallback ( Widget, XtPointer, XtPointer );
```

```
static void widthDragCallback ( Widget, XtPointer, XtPointer );
```

696

```
//----- Start editable code block: BbHistogram private
```

```
//----- End editable code block: BbHistogram private
```

```
};
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

/////////////////////////////////////////////////////////////////
//
// Header file for BbLConfig
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLConfigUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
/////////////////////////////////////////////////////////////////
#ifndef BBLCONFIG_H
#define BBLCONFIG_H
#include "BbLConfigUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "Patients.h"

//---- End editable code block: headers and declarations

//---- BbLConfig class declaration
class BbLConfig : public BbLConfigUI
{
public:
    BbLConfig ( const char *, Widget );
    BbLConfig ( const char * );
    ~BbLConfig();
    const char * className();

    static VkComponent *CreateBbLConfig( const char *name, Widget parent );

//---- Start editable code block: BbLConfig public
    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _patient_no, _study_no;

    void init0();
    Patients *init();
    void set_patients();
    void set_studies(int index_study);

//---- End editable code block: BbLConfig public

protected:

```

```
// These functions will be called as a result of calls
// registered in BbLConfigGUI

virtual void anatomy ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void patients ( Widget, XtPointer );

//----- Start editable code block: BbLConfig protected

//----- End editable code block: BbLConfig protected

private:

    static void* RegisterBbLConfigInterface();

    //----- Start editable code block: BbLConfig private

    //----- End editable code block: BbLConfig private

};
//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

#endif
```

```

//////////////////////////////////////
//
// Header file for BbLConfigUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLConfig is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLConfig files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
//////////////////////////////////////
#ifndef BBLCONFIGUI_H
#define BBLCONFIGUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class BbLConfigUI : public VkComponent
{
public:
    BbLConfigUI ( const char *, Widget );
    BbLConfigUI ( const char * );
    ~BbLConfigUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbLConfig public

    //--- End editable code block: BbLConfig public

protected:

    // Widgets created by this class

    Widget _bbLConfig;
    Widget _buttonAccept;
    Widget _scrolledListAnatomy;
    Widget _scrolledListPatients;
    Widget _scrolledWindowAnatomy;
    Widget _scrolledWindowPatients;

```



```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions 700
```

```
virtual void anatomy ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void patients ( Widget, XtPointer );

//---- Start editable code block: BbLConfig protected

//---- End editable code block: BbLConfig protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbLConfigUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void anatomyCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );
static void patientsCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbLConfig private
```

```
//---- End editable code block: BbLConfig private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbLPCMRA
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLPCMRAUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBLPCMRA_H
#define BBLPCMRA_H
#include "BbLPCMRAUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbLPCMRA class declaration

class BbLPCMRA : public BbLPCMRAUI
{
public:

    BbLPCMRA ( const char *, Widget );
    BbLPCMRA ( const char * );
    ~BbLPCMRA();
    const char * className();

    static VkComponent *CreateBbLPCMRA( const char *name, Widget parent );

//---- Start editable code block: BbLPCMRA public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

//---- End editable code block: BbLPCMRA public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbLPCMRAUI

    virtual void doButtonHideLocalizer ( Widget, XtPointer );
    virtual void doButtonShowLocalizer ( Widget, XtPointer );

//---- Start editable code block: BbLPCMRA protected

```

//---- End editable code block: BbLPCMRA protected

702

private:

static void* RegisterBbLPCMRAInterface();

//---- Start editable code block: BbLPCMRA private

//---- End editable code block: BbLPCMRA private

};

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```

////////////////////////////////////
//
// Header file for BbLPCMRAUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLPCMRA is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLPCMRA files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBLPCMRAUI_H
#define BBLPCMRAUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class BbLPCMRAUI : public VkComponent
{
public:

    BbLPCMRAUI ( const char *, Widget );
    BbLPCMRAUI ( const char * );
    ~BbLPCMRAUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbLPCMRA public

    //--- End editable code block: BbLPCMRA public

protected:

    // Widgets created by this class

    Widget _bbLPCMRA;
    Widget _buttonHideLocalizer;
    Widget _buttonShowLocalizer;

    // These virtual functions are called from the private callbacks (below)
    // Intended to be overridden in derived classes to define actions

```

```
virtual void doButtonHideLocalizer ( Widget, XtPointer );  
virtual void doButtonShowLocalizer ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLPCMRA protected
```

```
//---- End editable code block: BbLPCMRA protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbLPCMRAUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doButtonHideLocalizerCallback ( Widget, XtPointer, XtPointer );
```

```
static void doButtonShowLocalizerCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbLPCMRA private
```

```
//---- End editable code block: BbLPCMRA private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbHistogram.h

```

////////////////////////////////////
//
// Header file for BbLROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBLROI_H
#define BBLROI_H
#include "BbLROIUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbLROI class declaration

class BbLROI : public BbLROIUI
{
public:
    BbLROI ( const char *, Widget );
    BbLROI ( const char * );
    ~BbLROI();
    const char * className();

    static VkComponent *CreateBbLROI( const char *name, Widget parent );

    //---- Start editable code block: BbLROI public
    int _roi_mode;
    int _roi_type;
    int _roi_action;
    int _roi_color;

    void init();
    void initROI();
    void init2();
    void changeROI();
    void set_color();
    void set_type();

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    //---- End editable code block: BbLROI public

```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbLROIUI
```

```
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doButtonHide ( Widget, XtPointer );
virtual void doButtonShow ( Widget, XtPointer );
virtual void doOptionBlack ( Widget, XtPointer );
virtual void doOptionBlue ( Widget, XtPointer );
virtual void doOptionDraw ( Widget, XtPointer );
virtual void doOptionEllipse ( Widget, XtPointer );
virtual void doOptionEraseLeft ( Widget, XtPointer );
virtual void doOptionEraseRight ( Widget, XtPointer );
virtual void doOptionFreeHand ( Widget, XtPointer );
virtual void doOptionGreen ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionPolygon ( Widget, XtPointer );
virtual void doOptionRectangle ( Widget, XtPointer );
virtual void doOptionRed ( Widget, XtPointer );
virtual void doOptionWhite ( Widget, XtPointer );
virtual void doOptionYellow ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLROI protected
```

```
//---- End editable code block: BbLROI protected
```

private:

```
static void* RegisterBbLROIInterface();
```

```
//---- Start editable code block: BbLROI private
```

```
//---- End editable code block: BbLROI private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```



```

////////////////////////////////////
//
// Header file for BbLROIUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLROI is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLROI files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBLROIUI_H
#define BBLROIUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbLROIUI : public VkComponent
{
public:

    BbLROIUI ( const char *, Widget );
    BbLROIUI ( const char * );
    ~BbLROIUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbLROI public

    //--- End editable code block: BbLROI public

protected:

    // Widgets created by this class

    Widget _bbLROI;
    Widget _buttonAcceptArea;
    Widget _buttonHide;
    Widget _buttonShow;

```

```
VkOptionMenu *_optionMenu6;  
VkOptionMenu *_optionMenuColor;  
VkOptionMenu *_optionMenuROIType;
```

```
VkMenuItem *_optionBlack;  
VkMenuItem *_optionBlue;  
VkMenuItem *_optionDraw;  
VkMenuItem *_optionEllipse;  
VkMenuItem *_optionEraseLeft;  
VkMenuItem *_optionEraseRight;  
VkMenuItem *_optionFreeHand;  
VkMenuItem *_optionGreen;  
VkMenuItem *_optionModify;  
VkMenuItem *_optionPolygon;  
VkMenuItem *_optionRectangle;  
VkMenuItem *_optionRed;  
VkMenuItem *_optionWhite;  
VkMenuItem *_optionYellow;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doButtonAccept ( Widget, XtPointer );  
virtual void doButtonHide ( Widget, XtPointer );  
virtual void doButtonShow ( Widget, XtPointer );  
virtual void doOptionBlack ( Widget, XtPointer );  
virtual void doOptionBlue ( Widget, XtPointer );  
virtual void doOptionDraw ( Widget, XtPointer );  
virtual void doOptionEllipse ( Widget, XtPointer );  
virtual void doOptionEraseLeft ( Widget, XtPointer );  
virtual void doOptionEraseRight ( Widget, XtPointer );  
virtual void doOptionFreeHand ( Widget, XtPointer );  
virtual void doOptionGreen ( Widget, XtPointer );  
virtual void doOptionModify ( Widget, XtPointer );  
virtual void doOptionPolygon ( Widget, XtPointer );  
virtual void doOptionRectangle ( Widget, XtPointer );  
virtual void doOptionRed ( Widget, XtPointer );  
virtual void doOptionWhite ( Widget, XtPointer );  
virtual void doOptionYellow ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLROI protected
```

```
//---- End editable code block: BbLROI protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbLROIUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );  
static void doButtonHideCallback ( Widget, XtPointer, XtPointer );  
static void doButtonShowCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBlackCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBlueCallback ( Widget, XtPointer, XtPointer );  
static void doOptionDrawCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEllipseCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEraseLeftCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEraseRightCallback ( Widget, XtPointer, XtPointer );
```

```
static void doOptionFreeHandCallback ( Widget, XtPointer, XtPointer );
static void doOptionGripCallback ( Widget, XtPointer, XtPointer );
static void doOptionModifyCallback ( Widget, XtPointer, XtPointer );
static void doOptionPolygonCallback ( Widget, XtPointer, XtPointer );
static void doOptionRectangleCallback ( Widget, XtPointer, XtPointer );
static void doOptionRedCallback ( Widget, XtPointer, XtPointer );
static void doOptionWhiteCallback ( Widget, XtPointer, XtPointer );
static void doOptionYellowCallback ( Widget, XtPointer, XtPointer );
```

```
//----- Start editable code block: BbLROI private
```

```
//----- End editable code block: BbLROI private
```

```
};
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbLWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBLWAVEFORM_H
#define BBLWAVEFORM_H
#include "BbLWaveformUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbLWaveform class declaration

class BbLWaveform : public BbLWaveformUI
{
public:
    BbLWaveform ( const char *, Widget );
    BbLWaveform ( const char * );
    ~BbLWaveform();
    const char * className();

    static VkComponent *CreateBbLWaveform( const char *name, Widget parent );

//---- Start editable code block: BbLWaveform public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel); _current_vessel = 0}

    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);

//---- End editable code block: BbLWaveform public

protected:

```

```
// These functions will be called as a result of calls to  
// registered in BbLWaveformUI
```

704

```
virtual void doOptionASV ( Widget, XtPointer );  
virtual void doOptionArea ( Widget, XtPointer );  
virtual void doOptionBSV ( Widget, XtPointer );  
virtual void doOptionPSV ( Widget, XtPointer );  
virtual void doOptionVFR ( Widget, XtPointer );  
virtual void vessellWaveform ( Widget, XtPointer );
```

```
//----- Start editable code block: BbLWaveform protected
```

```
//----- End editable code block: BbLWaveform protected
```

```
private:
```

```
static void* RegisterBbLWaveformInterface();
```

```
//----- Start editable code block: BbLWaveform private
```

```
//----- End editable code block: BbLWaveform private
```

```
};  
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbLWaveformUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLWaveform is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLWaveform files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBLWAVEFORMUI_H
#define BBLWAVEFORMUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class BbLWaveformUI : public VkComponent
{
public:

    BbLWaveformUI ( const char *, Widget );
    BbLWaveformUI ( const char * );
    ~BbLWaveformUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbLWaveform public

    /*--- End editable code block: BbLWaveform public

protected:

    // Widgets created by this class

    Widget _bbLWaveform;
    Widget _labelCurrentNum1;
    Widget _labelMaxNum2;
    Widget _labelMinNum2;

```

```

Widget _labelUnit1;
Widget _scrolledList;
Widget _scrolledWindow1;

```

706

```

VkOptionMenu *_optionMenuFlow;

```

```

VkMenuItem *_optionASV;
VkMenuItem *_optionArea;
VkMenuItem *_optionBSV;
VkMenuItem *_optionPSV;
VkMenuItem *_optionVFR;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessellWaveform ( Widget, XtPointer );

```

```

//---- Start editable code block: BbLWaveform protected

```

```

//---- End editable code block: BbLWaveform protected

```

```

private:

```

```

// Array of default resources

```

```

static String _defaultBbLWaveformUIResources[];

```

```

// Callbacks to interface with Motif

```

```

static void doOptionASVCallback ( Widget, XtPointer, XtPointer );
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );
static void vessellWaveformCallback ( Widget, XtPointer, XtPointer );

```

```

//---- Start editable code block: BbLWaveform private

```

```

//---- End editable code block: BbLWaveform private

```

```

};
//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

```

#endif

```

User: meide
Host: phoenix
Class: phoenix
Job: BbLROI.h

707


```

////////////////////////////////////
//
// Header file for BbRHistogram
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRHistogramUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRHISTOGRAM_H
#define BBRHISTOGRAM_H
#include "BbRHistogramUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbRHistogram class declaration

class BbRHistogram : public BbRHistogramUI
{
public:
    BbRHistogram ( const char *, Widget );
    BbRHistogram ( const char * );
    ~BbRHistogram();
    const char * className();

    static VkComponent *CreateBbRHistogram( const char *name, Widget parent );

//---- Start editable code block: BbRHistogram public

    int _winWidth;
    int _winCenter;

    void init();

    void update(float, float);
    void update_lowhigh(float, float);

    void update_width(int);
    void update_center(int);

    void set_mapLabels();

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbRHistogramUI
```

```
virtual void AneurysmFlow ( Widget, XtPointer );
virtual void BlackFlow ( Widget, XtPointer );
virtual void WhiteFlow ( Widget, XtPointer );
virtual void abdomen2 ( Widget, XtPointer );
virtual void bone2 ( Widget, XtPointer );
virtual void centerDrag2 ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head2 ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung2 ( Widget, XtPointer );
virtual void mediastinum2 ( Widget, XtPointer );
virtual void spine2 ( Widget, XtPointer );
virtual void widthDrag2 ( Widget, XtPointer );
```

//---- Start editable code block: BbRHistogram protected

//---- End editable code block: BbRHistogram protected

private:

static void* RegisterBbRHistogramInterface();

//---- Start editable code block: BbRHistogram private

//---- End editable code block: BbRHistogram private

```
};
//---- Start editable code block: End of generated code
```

//---- End editable code block: End of generated code

#endif

```

////////////////////////////////////
//
// Header file for BbRHistogramUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRHistogram is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRHistogram files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRHISTOGRAMUI_H
#define BBRHISTOGRAMUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class BbRHistogramUI : public VkComponent
{
public:

    BbRHistogramUI ( const char *, Widget );
    BbRHistogramUI ( const char * );
    ~BbRHistogramUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbRHistogram public

    void create_mapLabels();
    void remove_mapLabels();

    Widget _bbRHistogram;
    Widget _dialCenter2;
    Widget _dialWidth2;
    Widget _labelLHistoHigh2;
    Widget _labelLHistoLow2;
    Widget _labelLHistoMax2;
    Widget _labelLHistoMin2;
    Widget _labelMap1;

```

```
Widget _labelMap2;
Widget _labelMap3;
Widget _labelMap4;
Widget _labelMap5;
Widget _labelMap6;
Widget _labelMap7;
Widget _labelMap8;
Widget _labelMap9;
```

```
//---- End editable code block: BbRHistogram public
```

```
protected:
```

```
// Widgets created by this class
```

```
VkOptionMenu *_optionMenuLHistogram21;
VkOptionMenu *_optionMenuRhists;
```

```
VkMenuItem *_optionAbdomen1;
VkMenuItem *_optionAneurysmFlow;
VkMenuItem *_optionBlackFlow;
VkMenuItem *_optionBone1;
VkMenuItem *_optionCoarse;
VkMenuItem *_optionFine;
VkMenuItem *_optionHead1;
VkMenuItem *_optionLung1;
VkMenuItem *_optionMapping;
VkMenuItem *_optionMediaStinum1;
VkMenuItem *_optionROI1;
VkMenuItem *_optionSpine1;
VkMenuItem *_optionUpdate;
VkMenuItem *_optionWhiteFlow;
```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions
```

```
virtual void AneurysmFlow ( Widget, XtPointer );
virtual void BlackFlow ( Widget, XtPointer );
virtual void WhiteFlow ( Widget, XtPointer );
virtual void abdomen2 ( Widget, XtPointer );
virtual void bone2 ( Widget, XtPointer );
virtual void centerDrag2 ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head2 ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung2 ( Widget, XtPointer );
virtual void mediastinum2 ( Widget, XtPointer );
virtual void spine2 ( Widget, XtPointer );
virtual void widthDrag2 ( Widget, XtPointer );
```

```
//---- Start editable code block: BbRHistogram protected
```

```
//---- End editable code block: BbRHistogram protected
```

```
// Array of default resources
```

```
static String      _defaultBbRHistogramUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void AneurysmFlowCallback ( Widget, XtPointer, XtPointer );
static void BlackFlowCallback ( Widget, XtPointer, XtPointer );
static void WhiteFlowCallback ( Widget, XtPointer, XtPointer );
static void abdomen2Callback ( Widget, XtPointer, XtPointer );
static void bone2Callback ( Widget, XtPointer, XtPointer );
static void centerDrag2Callback ( Widget, XtPointer, XtPointer );
static void doOptionCoarseCallback ( Widget, XtPointer, XtPointer );
static void doOptionFineCallback ( Widget, XtPointer, XtPointer );
static void doOptionMappingCallback ( Widget, XtPointer, XtPointer );
static void doOptionROICallback ( Widget, XtPointer, XtPointer );
static void doOptionUpdateCallback ( Widget, XtPointer, XtPointer );
static void head2Callback ( Widget, XtPointer, XtPointer );
static void highChgCallback ( Widget, XtPointer, XtPointer );
static void lowChgCallback ( Widget, XtPointer, XtPointer );
static void lung2Callback ( Widget, XtPointer, XtPointer );
static void mediastinum2Callback ( Widget, XtPointer, XtPointer );
static void spine2Callback ( Widget, XtPointer, XtPointer );
static void widthDrag2Callback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbRHistogram private
```

```
//---- End editable code block: BbRHistogram private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRROI_H
#define BBRROI_H
#include "BbRROIUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbRROI class declaration

class BbRROI : public BbRROIUI
{
public:
    BbRROI ( const char *, Widget );
    BbRROI ( const char * );
    ~BbRROI();
    const char * className();

    static VkComponent *CreateBbRROI( const char *name, Widget parent );

//---- Start editable code block: BbRROI public
    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _ROI_NO;
    void draw_ROI();
    void draw_AllROI(int);
    void show_current(int i);
    void show_total(int i);
    void modify();
    void add_AllROI(int, int);

    int modify(char *name);

    int _mode;
    int _roi_nn;
    int _frame;

```

```
void set_list();
void draw_ROINeighbor
```

714

```
//----- End editable code block: BbRROI public
```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbRROIUI
```

```
virtual void NextNeighbor ( Widget, XtPointer );
virtual void PrevROI ( Widget, XtPointer );
virtual void ROIName ( Widget, XtPointer );
virtual void doButtonAcceptROI ( Widget, XtPointer );
virtual void doButtonRemove ( Widget, XtPointer );
virtual void doButtonSaveROI ( Widget, XtPointer );
virtual void doOptionBackFlow ( Widget, XtPointer );
virtual void doOptionHide ( Widget, XtPointer );
virtual void doOptionHideNeighbor ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionModify3D ( Widget, XtPointer );
virtual void doOptionOpenROI ( Widget, XtPointer );
virtual void doOptionROIFlow ( Widget, XtPointer );
virtual void doOptionSave3D ( Widget, XtPointer );
virtual void doOptionShow ( Widget, XtPointer );
virtual void doOptionShow3D ( Widget, XtPointer );
virtual void doOptionShowAll ( Widget, XtPointer );
virtual void doOptionShowAllNeighbor ( Widget, XtPointer );
virtual void doOptionShowNeighbor ( Widget, XtPointer );
virtual void rois ( Widget, XtPointer );
```

```
//----- Start editable code block: BbRROI protected
```

```
//----- End editable code block: BbRROI protected
```

private:

```
static void* RegisterBbRROIInterface();
```

```
//----- Start editable code block: BbRROI private
```

```
//----- End editable code block: BbRROI private
```

```
};
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbRHistogram.h

715


```

////////////////////////////////////
//
// Header file for BbRROIUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRROI is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRROI files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRROIUI_H
#define BBRROIUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItemToggle;
class BbRROIUI : public VkComponent
{
public:
    BbRROIUI ( const char *, Widget );
    BbRROIUI ( const char * );
    ~BbRROIUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbRROI public

    /*--- End editable code block: BbRROI public

protected:

    // Widgets created by this class

    Widget _arrow1;
    Widget _arrowPrevROI;
    Widget _bbRROI;
    Widget _buttonAcceptROI;

```

```

Widget _buttonRemoveROI;
Widget _buttonSaveROI;
Widget _labelNumCurr;
Widget _labelNumROI;
Widget _labelROIName;
Widget _scrolledListROIS;
Widget _scrolledWindow6;
Widget _separator3;
Widget _textfieldROIName;

```

```

VkOptionMenu *_optionMenu5;
VkOptionMenu *_optionMenu7;
VkOptionMenu *_optionMenu8;

```

```

VkMenuItem *_optionBackFlow;
VkMenuItem *_optionHide;
VkMenuItem *_optionHideNeighbor;
VkMenuItem *_optionModify3D;
VkMenuItem *_optionModifyROI;
VkMenuItem *_optionOpenROI;
VkMenuItem *_optionROIFlow;
VkMenuItem *_optionSave3D;
VkMenuItem *_optionShow;
VkMenuItem *_optionShow3D;
VkMenuItem *_optionShowAll;
VkMenuItem *_optionShowAllNeighbor;
VkMenuItem *_optionShowNeighbor;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void NextNeighbor ( Widget, XtPointer );
virtual void PrevROI ( Widget, XtPointer );
virtual void ROIName ( Widget, XtPointer );
virtual void doButtonAcceptROI ( Widget, XtPointer );
virtual void doButtonRemove ( Widget, XtPointer );
virtual void doButtonSaveROI ( Widget, XtPointer );
virtual void doOptionBackFlow ( Widget, XtPointer );
virtual void doOptionHide ( Widget, XtPointer );
virtual void doOptionHideNeighbor ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionModify3D ( Widget, XtPointer );
virtual void doOptionOpenROI ( Widget, XtPointer );
virtual void doOptionROIFlow ( Widget, XtPointer );
virtual void doOptionSave3D ( Widget, XtPointer );
virtual void doOptionShow ( Widget, XtPointer );
virtual void doOptionShow3D ( Widget, XtPointer );
virtual void doOptionShowAll ( Widget, XtPointer );
virtual void doOptionShowAllNeighbor ( Widget, XtPointer );
virtual void doOptionShowNeighbor ( Widget, XtPointer );
virtual void rois ( Widget, XtPointer );

```

```

//---- Start editable code block: BbRROI protected

```

```

//---- End editable code block: BbRROI protected

```

```

private:

```

```

// Array of default resources

```

```

static String _defaultBbRROIUIResources[];

```

```
static void NextNeighborCallback ( Widget, XtPointer, XtPointer );
static void PrevROIDCallback ( Widget, XtPointer, XtPointer );
static void ROINameCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptROIDCallback ( Widget, XtPointer, XtPointer );
static void doButtonRemoveCallback ( Widget, XtPointer, XtPointer );
static void doButtonSaveROIDCallback ( Widget, XtPointer, XtPointer );
static void doOptionBackFlowCallback ( Widget, XtPointer, XtPointer );
static void doOptionHideCallback ( Widget, XtPointer, XtPointer );
static void doOptionHideNeighborCallback ( Widget, XtPointer, XtPointer );
static void doOptionModifyCallback ( Widget, XtPointer, XtPointer );
static void doOptionModify3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionOpenROIDCallback ( Widget, XtPointer, XtPointer );
static void doOptionROIFlowCallback ( Widget, XtPointer, XtPointer );
static void doOptionSave3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowCallback ( Widget, XtPointer, XtPointer );
static void doOptionShow3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowAllCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowAllNeighborCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowNeighborCallback ( Widget, XtPointer, XtPointer );
static void roisCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbRROI private
```

```
//---- End editable code block: BbRROI private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRTable
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRTableUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRTABLE_H
#define BBRTABLE_H
#include "BbRTableUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbRTable class declaration

class BbRTable : public BbRTableUI
{
public:
    BbRTable ( const char *, Widget );
    BbRTable ( const char * );
    ~BbRTable();
    const char * className();

    static VkComponent *CreateBbRTable( const char *name, Widget parent );

//---- Start editable code block: BbRTable public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel3); _current_vessel =

    void show_info();
    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);
    void set_list(int num, float *x);

//---- End editable code block: BbRTable public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in BbRTableUI
```

```
virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessel ( Widget, XtPointer );
virtual void vesselRTable ( Widget, XtPointer );
```

```
//---- Start editable code block: BbRTable protected
```

```
//---- End editable code block: BbRTable protected
```

```
private:
```

```
static void* RegisterBbRTableInterface();
```

```
//---- Start editable code block: BbRTable private
```

```
//---- End editable code block: BbRTable private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRTableUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRTable is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRTable files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRTABLEUI_H
#define BBRTABLEUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItemToggle;
class BbRTableUI : public VkComponent
{
public:
    BbRTableUI ( const char *, Widget );
    BbRTableUI ( const char * );
    ~BbRTableUI();
    void create ( Widget );
    const char * className();

//---- Start editable code block: BbRTable public

//---- End editable code block: BbRTable public

protected:

// Widgets created by this class
Widget _bbRTable;
Widget _labelAverage1;
Widget _labelAverageNum1;
Widget _labelHeartRate1;

```

```

Widget _labelHeartRateNum1;
Widget _labelMax1;
Widget _labelMaxNum1;
Widget _labelMin1;
Widget _labelMinNum1;
Widget _labelUnits;
Widget _labelUnitsNum;
Widget _labelVolume1;
Widget _labelVolumeNum1;
Widget _scrolledListVessel1;
Widget _scrolledListVessel3;
Widget _scrolledWindow2;
Widget _scrolledWindow4;

```

```
VkOptionMenu *_optionMenuFlow2;
```

```

VkMenuItem *_optionASV2;
VkMenuItem *_optionArea2;
VkMenuItem *_optionBSV2;
VkMenuItem *_optionPSV2;
VkMenuItem *_optionVFR2;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessel ( Widget, XtPointer );
virtual void vesselRTable ( Widget, XtPointer );

```

```
//---- Start editable code block: BbRTable protected
```

```
//---- End editable code block: BbRTable protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbRTableUIResources[];
```

```
// Callbacks to interface with Motif
```

```

static void doOptionASVCallback ( Widget, XtPointer, XtPointer );
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );
static void vesselCallback ( Widget, XtPointer, XtPointer );
static void vesselRTableCallback ( Widget, XtPointer, XtPointer );

```

```
//---- Start editable code block: BbRTable private
```

```
//---- End editable code block: BbRTable private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

#endif

723


```

/////////////////////////////////////////////////////////////////
//
// Header file for BbRWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
/////////////////////////////////////////////////////////////////
#ifndef BBRWAVEFORM_H
#define BBRWAVEFORM_H
#include "BbRWaveformUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbRWaveform class declaration

class BbRWaveform : public BbRWaveformUI
{
public:
    BbRWaveform ( const char *, Widget );
    BbRWaveform ( const char * );
    ~BbRWaveform();
    const char * className();

    static VkComponent *CreateBbRWaveform( const char *name, Widget parent );

//---- Start editable code block: BbRWaveform public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel2); _current_vessel =

    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);

//---- End editable code block: BbRWaveform public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in BbRWaveformUI
```

725

```
virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vesselRWaveform ( Widget, XtPointer );
```

```
//----- Start editable code block: BbRWaveform protected
```

```
//----- End editable code block: BbRWaveform protected
```

```
private:
```

```
static void* RegisterBbRWaveformInterface();
```

```
//----- Start editable code block: BbRWaveform private
```

```
//----- End editable code block: BbRWaveform private
```

```
};
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRWaveformUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRWaveform is instantiated
//
// To extend or alter the behavior of this class, you should
-// modify the BbRWaveform files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRWAVEFORMUI_H
#define BBRWAVEFORMUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class VkMenuItem;
class BbRWaveformUI : public VkComponent
{
public:
    BbRWaveformUI ( const char *, Widget );
    BbRWaveformUI ( const char * );
    ~BbRWaveformUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbRWaveform public

    //--- End editable code block: BbRWaveform public

protected:

    // Widgets created by this class

    Widget _bbRWaveform;
    Widget _labelCurrentNum;
    Widget _labelMaxNum;
    Widget _labelMinNum;

```

```
Widget _labelUnit;  
Widget _scrolledListWidget2;  
Widget _scrolledWindow;
```

727

```
VkOptionMenu *_optionMenuFlow1;
```

```
VkMenuItem *_optionASV1;  
VkMenuItem *_optionArea1;  
VkMenuItem *_optionBSV1;  
VkMenuItem *_optionPSV1;  
VkMenuItem *_optionVFR1;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doOptionASV ( Widget, XtPointer );  
virtual void doOptionArea ( Widget, XtPointer );  
virtual void doOptionBSV ( Widget, XtPointer );  
virtual void doOptionPSV ( Widget, XtPointer );  
virtual void doOptionVFR ( Widget, XtPointer );  
virtual void vesselRWaveform ( Widget, XtPointer );
```

```
//---- Start editable code block: BbRWaveform protected
```

```
//---- End editable code block: BbRWaveform protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbRWaveformUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doOptionASVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );  
static void vesselRWaveformCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbRWaveform private
```

```
//---- End editable code block: BbRWaveform private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, Bb is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the Bb files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBUI_H
#define BBUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "Flow.h"

//---- End editable code block: headers and declarations

// Externally defined classes referenced by this class:

class DeckLTabbedDeck;
class DeckRTabbedDeck;

class VkOptionMenu;
class VkMenuItem;
class VkMenuItemToggle;
class VkMenuItem;
class BbUI : public VkComponent
{
public:
    BbUI ( const char *, Widget );
    BbUI ( const char * );
    ~BbUI();
    void create ( Widget );
    const char * className();

//---- Start editable code block: Bb public

    ObjectManager *_objMag;
    void set(class ObjectManager *);

    void init();

```

```

void init_patient();

Flow * init_flow(int m);
void remove_flow();

Widget _labelImgNumber;

//---- End editable code block: Bb public

```

protected:

```
// Classes created by this class
```

```

class DeckRTabbedDeck *_deckR;
class DeckLTabbedDeck *_deckL;

```

```
// Widgets created by this class
```

```

Widget _arrowNext;
Widget _arrowPrev;
Widget _bb;
//Widget _labelImgNumber;
Widget _separatorBottom;
Widget _separatorMiddle;
Widget _separatorTop;

```

```

VkOptionMenu *_optionMenuAnimate;
VkOptionMenu *_optionMenuPCMR;
VkOptionMenu *_optionMenuSelect;
VkOptionMenu *_optionMenuSpace;
VkOptionMenu *_optionMenuVisual;

```

```

VkMenuItem *_option3D;
VkMenuItem *_optionAnimate;
VkMenuItem *_optionColor2D;
VkMenuItem *_optionGray2D;
VkMenuItem *_optionMagnitude;
VkMenuItem *_optionNewAnimate;
VkMenuItem *_optionOther;
VkMenuItem *_optionPhase;
VkMenuItem *_optionROI;
VkMenuItem *_optionReference;
VkMenuItem *_optionSimple;
VkMenuItem *_optionSpline;
VkMenuItem *_optionStopAnimate;
VkMenuItem *_optionVelocity;
VkMenuItem *_optionWhole;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void Next ( Widget, XtPointer );
virtual void Prev ( Widget, XtPointer );
virtual void doOption3D ( Widget, XtPointer );
virtual void doOptionAnimate ( Widget, XtPointer );
virtual void doOptionColor2D ( Widget, XtPointer );
virtual void doOptionGray2D ( Widget, XtPointer );
virtual void doOptionMagnitude ( Widget, XtPointer );
virtual void doOptionNewAnimate ( Widget, XtPointer );
virtual void doOptionOther ( Widget, XtPointer );
virtual void doOptionPhase ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionReference ( Widget, XtPointer );

```

```

virtual void doOptionSimple ( Widget, XtPointer );
virtual void doOptionSpline ( Widget, XtPointer );
virtual void doOptionSplineAnimate ( Widget, XtPointer );
virtual void doOptionVelocity ( Widget, XtPointer );
virtual void doOptionWhole ( Widget, XtPointer );

```

```
//---- Start editable code block: Bb protected
```

```
//---- End editable code block: Bb protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbUIResources[];
```

```
// Callbacks to interface with Motif
```

```

static void NextCallback ( Widget, XtPointer, XtPointer );
static void PrevCallback ( Widget, XtPointer, XtPointer );
static void doOption3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionColor2DCallback ( Widget, XtPointer, XtPointer );
static void doOptionGray2DCallback ( Widget, XtPointer, XtPointer );
static void doOptionMagnitudeCallback ( Widget, XtPointer, XtPointer );
static void doOptionNewAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionOtherCallback ( Widget, XtPointer, XtPointer );
static void doOptionPhaseCallback ( Widget, XtPointer, XtPointer );
static void doOptionROICallback ( Widget, XtPointer, XtPointer );
static void doOptionReferenceCallback ( Widget, XtPointer, XtPointer );
static void doOptionSimpleCallback ( Widget, XtPointer, XtPointer );
static void doOptionSplineCallback ( Widget, XtPointer, XtPointer );
static void doOptionStopAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionVelocityCallback ( Widget, XtPointer, XtPointer );
static void doOptionWholeCallback ( Widget, XtPointer, XtPointer );

```

```
//---- Start editable code block: Bb private
```

```
//---- End editable code block: Bb private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbVelocity
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVelocityUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBVELOCITY_H
#define BBVELOCITY_H
#include "BbVelocityUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbVelocity class declaration

class BbVelocity : public BbVelocityUI
{
public:
    BbVelocity ( const char *, Widget );
    BbVelocity ( const char * );
    ~BbVelocity();
    const char * className();

    static VkComponent *CreateBbVelocity( const char *name, Widget parent );

//---- Start editable code block: BbVelocity public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

//---- End editable code block: BbVelocity public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbVelocityUI

    virtual void Ratio ( Widget, XtPointer );
    virtual void doOption100 ( Widget, XtPointer );
    virtual void doOption25 ( Widget, XtPointer );

```



```
virtual void doOption50 ( Widget, XtPointer );
virtual void doOption7 ( Widget, XtPointer );
virtual void doOptionA ( Widget, XtPointer );
virtual void doOptionFlowMasked ( Widget, XtPointer );
virtual void doOptionNone ( Widget, XtPointer );
virtual void doOptionROIMasked ( Widget, XtPointer );
virtual void threshMag ( Widget, XtPointer );
virtual void threshNeg ( Widget, XtPointer );
virtual void threshPos ( Widget, XtPointer );
```

```
//---- Start editable code block: BbVelocity protected
```

```
//---- End editable code block: BbVelocity protected
```

```
private:
```

```
static void* RegisterBbVelocityInterface();
```

```
//---- Start editable code block: BbVelocity private
```

```
//---- End editable code block: BbVelocity private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbRROIUI.h

733

```

////////////////////////////////////
//
// Header file for BbVelocityUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbVelocity is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbVelocity files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBVELOCITYUI_H
#define BBVELOCITYUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class BbVelocityUI : public VkComponent
{
public:
    BbVelocityUI ( const char *, Widget );
    BbVelocityUI ( const char * );
    ~BbVelocityUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbVelocity public

    /*--- End editable code block: BbVelocity public

protected:

    // Widgets created by this class

    Widget _bbVelocity;
    Widget _labelMagThresh;
    Widget _labelNegThresh;
    Widget _labelPhase2Velocity;

```

```
Widget _labelPosThresh;
Widget _textfieldMagThresh;
Widget _textfieldNegThresh;
Widget _textfieldPosThresh;
Widget _textfieldRatio;
```

```
VkOptionMenu *_optionMenu4;
VkOptionMenu *_optionMenuVelocityMethod;
```

```
VkMenuItem *_option1;
VkMenuItem *_option100;
VkMenuItem *_option25;
VkMenuItem *_option75;
VkMenuItem *_optionAsIs;
VkMenuItem *_optionFlowMasked;
VkMenuItem *_optionNone;
VkMenuItem *_optionROIMasked;
```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions
```

```
virtual void Ratio ( Widget, XtPointer );
virtual void doOption100 ( Widget, XtPointer );
virtual void doOption25 ( Widget, XtPointer );
virtual void doOption50 ( Widget, XtPointer );
virtual void doOption75 ( Widget, XtPointer );
virtual void doOptionAsIs ( Widget, XtPointer );
virtual void doOptionFlowMasked ( Widget, XtPointer );
virtual void doOptionNone ( Widget, XtPointer );
virtual void doOptionROIMasked ( Widget, XtPointer );
virtual void threshMag ( Widget, XtPointer );
virtual void threshNeg ( Widget, XtPointer );
virtual void threshPos ( Widget, XtPointer );
```

```
//---- Start editable code block: BbVelocity protected
```

```
//---- End editable code block: BbVelocity protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbVelocityUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void RatioCallback ( Widget, XtPointer, XtPointer );
static void doOption100Callback ( Widget, XtPointer, XtPointer );
static void doOption25Callback ( Widget, XtPointer, XtPointer );
static void doOption50Callback ( Widget, XtPointer, XtPointer );
static void doOption75Callback ( Widget, XtPointer, XtPointer );
static void doOptionAsIsCallback ( Widget, XtPointer, XtPointer );
static void doOptionFlowMaskedCallback ( Widget, XtPointer, XtPointer );
static void doOptionNoneCallback ( Widget, XtPointer, XtPointer );
static void doOptionROIMaskedCallback ( Widget, XtPointer, XtPointer );
static void threshMagCallback ( Widget, XtPointer, XtPointer );
static void threshNegCallback ( Widget, XtPointer, XtPointer );
static void threshPosCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbVelocity private
```

```
//---- End editable code block: BbVelocity private  
};  
//---- Start editable code block: End of generated code
```

736

```
//---- End editable code block: End of generated code  
#endif
```

```

////////////////////////////////////
//
// Header file for BbVisual
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVisualUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBVISUAL_H
#define BBVISUAL_H
#include "BbVisualUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/TextF.h>
#include <Vk/VkComponent.h>
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbVisual class declaration

class BbVisual : public BbVisualUI
{
public:

    BbVisual ( const char *, Widget );
    BbVisual ( const char * );
    ~BbVisual();
    const char * className();

    static VkComponent *CreateBbVisual( const char *name, Widget parent );

    //---- Start editable code block: BbVisual public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void accept();
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel4);}

    void set_flowdir(int);
    void add_flow(char *);

    char *get_patient() { return XmTextFieldGetString(_textfieldName);}
    char *get_vessel() { return XmTextFieldGetString(_textfieldVessel);}
    char *get_date() { return XmTextFieldGetString(_textfieldDate);}
    char *get_remark() { return XmTextFieldGetString(_textfieldRemark);}

    void set_info(char *name, char *studyDate, char *remark);

```

```

void setVessel(char *vesselomy);
void set_Path(char *p);
char *get_basePath();

//void update_histo(HistoView *his, int w, int h, short **img);

//---- End editable code block: BbVisual public

```

protected:

```

// These functions will be called as a result of callbacks
// registered in BbVisualUI

```

```

virtual void Vessel ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doVeLICANeck ( Widget, XtPointer );
virtual void doVesBAdown ( Widget, XtPointer );
virtual void doVesBAup ( Widget, XtPointer );
virtual void doVesLACA ( Widget, XtPointer );
virtual void doVesLCCA ( Widget, XtPointer );
virtual void doVesLECA ( Widget, XtPointer );
virtual void doVesLICAItra ( Widget, XtPointer );
virtual void doVesLMCA ( Widget, XtPointer );
virtual void doVesLVA ( Widget, XtPointer );
virtual void doVesNew ( Widget, XtPointer );
virtual void doVesRACA ( Widget, XtPointer );
virtual void doVesRCCA ( Widget, XtPointer );
virtual void doVesRECA ( Widget, XtPointer );
virtual void doVesRICAItra ( Widget, XtPointer );
virtual void doVesRICANeck ( Widget, XtPointer );
virtual void doVesRMCA ( Widget, XtPointer );
virtual void doVesRVA ( Widget, XtPointer );
virtual void setToggleFlowNeg ( Widget, XtPointer );
virtual void setToggleFlowNeutral ( Widget, XtPointer );
virtual void setToggleFlowPos ( Widget, XtPointer );
virtual void userName ( Widget, XtPointer );
virtual void vesselRUser ( Widget, XtPointer );

```

```

//---- Start editable code block: BbVisual protected

```

```

//---- End editable code block: BbVisual protected

```

private:

```

static void* RegisterBbVisualInterface();

```

```

//---- Start editable code block: BbVisual private

```

```

//---- End editable code block: BbVisual private

```

```

};
//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

```

#endif

```



```

////////////////////////////////////
//
// Header file for BbVisualUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbVisual is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbVisual files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBVISUALUI_H
#define BBVISUALUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class VkMenuItem;
class BbVisualUI : public VkComponent
{
public:

    BbVisualUI ( const char *, Widget );
    BbVisualUI ( const char * );
    ~BbVisualUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbVisual public

    /*--- End editable code block: BbVisual public

protected:

    // Widgets created by this class

    Widget _bbVisual;
    Widget _buttonAcceptUser;
    Widget _labelDate;
    Widget _labelDescription;

```

```

Widget _labelFlowDir;
Widget _labelVessel;
Widget _labelname;
Widget _radioboxFlowDir;
Widget _scrolledListVessel4;
Widget _scrolledWindow5;
Widget _textfieldDate;
Widget _textfieldName;
Widget _textfieldRemark;
Widget _textfieldVessel;
Widget _toggleFlowNeg;
Widget _toggleFlowNeutral;
Widget _toggleFlowPos;

```

```

VkOptionMenu *_optionMenu17;

```

```

VkMenuItem *_separator5;
VkMenuItem *_separator6;
VkMenuItem *_separator7;
VkMenuItem *_separator8;
VkMenuItem *_vesBAdown;
VkMenuItem *_vesBAup;
VkMenuItem *_vesLACA;
VkMenuItem *_vesLCCA;
VkMenuItem *_vesLECA;
VkMenuItem *_vesLICAItra;
VkMenuItem *_vesLICANeck;
VkMenuItem *_vesLMCA;
VkMenuItem *_vesLVA;
VkMenuItem *_vesNew;
VkMenuItem *_vesRACA;
VkMenuItem *_vesRCCA;
VkMenuItem *_vesRECA;
VkMenuItem *_vesRICAItra;
VkMenuItem *_vesRICANeck;
VkMenuItem *_vesRMCA;
VkMenuItem *_vesRVA;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void Vessel ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doVeLICANeck ( Widget, XtPointer );
virtual void doVesBAdown ( Widget, XtPointer );
virtual void doVesBAup ( Widget, XtPointer );
virtual void doVesLACA ( Widget, XtPointer );
virtual void doVesLCCA ( Widget, XtPointer );
virtual void doVesLECA ( Widget, XtPointer );
virtual void doVesLICAItra ( Widget, XtPointer );
virtual void doVesLMCA ( Widget, XtPointer );
virtual void doVesLVA ( Widget, XtPointer );
virtual void doVesNew ( Widget, XtPointer );
virtual void doVesRACA ( Widget, XtPointer );
virtual void doVesRCCA ( Widget, XtPointer );
virtual void doVesRECA ( Widget, XtPointer );
virtual void doVesRICAItra ( Widget, XtPointer );
virtual void doVesRICANeck ( Widget, XtPointer );
virtual void doVesRMCA ( Widget, XtPointer );
virtual void doVesRVA ( Widget, XtPointer );
virtual void setToggleFlowNeg ( Widget, XtPointer );
virtual void setToggleFlowNeutral ( Widget, XtPointer );
virtual void setToggleFlowPos ( Widget, XtPointer );
virtual void userName ( Widget, XtPointer );
virtual void vesselRUser ( Widget, XtPointer );

```



```
//---- Start editable code block: BbVisual protected
```

743

```
//---- End editable code block: BbVisual protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbVisualUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void VesselCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );
static void doVeLICANeckCallback ( Widget, XtPointer, XtPointer );
static void doVesBADownCallback ( Widget, XtPointer, XtPointer );
static void doVesBAupCallback ( Widget, XtPointer, XtPointer );
static void doVesLACACallback ( Widget, XtPointer, XtPointer );
static void doVesLCCACallback ( Widget, XtPointer, XtPointer );
static void doVesLECACallback ( Widget, XtPointer, XtPointer );
static void doVesLICAIntraCallback ( Widget, XtPointer, XtPointer );
static void doVesLMCACallback ( Widget, XtPointer, XtPointer );
static void doVesLVACallback ( Widget, XtPointer, XtPointer );
static void doVesNewCallback ( Widget, XtPointer, XtPointer );
static void doVesRACACallback ( Widget, XtPointer, XtPointer );
static void doVesRCCACallback ( Widget, XtPointer, XtPointer );
static void doVesRECCACallback ( Widget, XtPointer, XtPointer );
static void doVesRICAItraCallback ( Widget, XtPointer, XtPointer );
static void doVesRICANeckCallback ( Widget, XtPointer, XtPointer );
static void doVesRMCACallback ( Widget, XtPointer, XtPointer );
static void doVesRVACallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowNegCallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowNeutralCallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowPosCallback ( Widget, XtPointer, XtPointer );
static void userNameCallback ( Widget, XtPointer, XtPointer );
static void vesselRUserCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbVisual private
```

```
//---- End editable code block: BbVisual private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for DeckLTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef DECKLTABBEDDECK_H
#define DECKLTABBEDDECK_H
#include <Vk/VkTabbedDeck.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- DeckLTabbedDeck class declaration

class DeckLTabbedDeck : public VkTabbedDeck
{
public:
    DeckLTabbedDeck ( const char *, Widget );
    DeckLTabbedDeck ( const char * );
    ~DeckLTabbedDeck();
    const char * className();

    static VkComponent *CreateDeckLTabbedDeck( const char *name, Widget parent );

//---- Start editable code block: DeckLTabbedDeck public
    void set(class ObjectManager *);

//---- End editable code block: DeckLTabbedDeck public

protected:

    // Classes created by this class

    class BbHistogram *_bbHistogram;
    class BbDisplay *_bbDisplay;
    class BbLROI *_bbLROI;
    class BbLConfig *_bbLConfig;
    class BbLPCMRA *_bbLPCMRA;
    class BbLWaveform *_bbLWaveform;
    class BbDetail *_bbDetail;
    class BbLConfigNew *_bbLConfigNew;

    // Widgets created by this class

```

```
//---- Start editable code block: DeckLTabbedDeck protected
```

```
//---- End editable code block: DeckLTabbedDeck protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultDeckLTabbedDeckResources[];
```

```
//---- Start editable code block: DeckLTabbedDeck private
```

```
//---- End editable code block: DeckLTabbedDeck private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for DeckRTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef DECKRTABBEDDECK_H
#define DECKRTABBEDDECK_H
#include <Vk/VkTabbedDeck.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- DeckRTabbedDeck class declaration

class DeckRTabbedDeck : public VkTabbedDeck
{
public:
    DeckRTabbedDeck ( const char *, Widget );
    DeckRTabbedDeck ( const char * );
    ~DeckRTabbedDeck();
    const char * className();

    static VkComponent *CreateDeckRTabbedDeck( const char *name, Widget parent );

//---- Start editable code block: DeckRTabbedDeck public

    void set(class ObjectManager *);

//---- End editable code block: DeckRTabbedDeck public

protected:

    // Classes created by this class

    class BbRHistogram *_bbRHistogram;
    class BbFlow *_bbFlow;
    class BbVisual *_bbVisual;
    class BbFormat *_bbFormat;
    class BbFlow3D *_bbFlow3D;
    class Bb3DLocalizer *_bb3DLocalizer;
    class BbVelocity *_bbVelocity;
    class Bb3D *_bb3D;
    class BbAnimation *_bbAnimation;
    class BbRROI *_bbRROI;

```

```
class BbRWaveform *_bbRWaveform;  
class BbRTable *_bbRT;
```

747

```
// Widgets created by this class
```

```
Widget _deckR;
```

```
//----- Start editable code block: DeckRTabbedDeck protected
```

```
//----- End editable code block: DeckRTabbedDeck protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultDeckRTabbedDeckResources[];
```

```
//----- Start editable code block: DeckRTabbedDeck private
```

```
//----- End editable code block: DeckRTabbedDeck private
```

```
};  
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```


User: meide
Host: phoenix
Class: phoenix
Job: BbVelocityUI.h

748

```

////////////////////////////////////
//
// Header file for VkwindowMainWindow
//
// This class is a subclass of VkWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
////////////////////////////////////
#ifndef VKWINDOWMAINWINDOW_H
#define VKWINDOWMAINWINDOW_H
#include <Vk/VkWindow.h>

class VkMenuItem;
class VkMenuToggle;
class VkMenuConfirmFirstAction;
class VkSubMenu;
class VkRadioSubMenu;

/*--- Start editable code block: headers and declarations
/*--- End editable code block: headers and declarations

/*--- VkwindowMainWindow class declaration
class VkwindowMainWindow: public VkWindow {
public:
    VkwindowMainWindow( const char * name,
                        ArgList args = NULL,
                        Cardinal argCount = 0 );
    ~VkwindowMainWindow();
    const char *className();
    virtual Boolean okToQuit();

/*--- Start editable code block: VkwindowMainWindow public
/*--- End editable code block: VkwindowMainWindow public

protected:

    // Classes created by this class

    class Bb *_bb;

    // Widgets created by this class

    // Menu items created by this class
    VkSubMenu *_filePane;
    VkMenuItem *_newButton;
    VkMenuItem *_openButton;

```

```

VkMenuItem *_saveButton;
VkMenuItem *_saveasButton;
VkMenuItem *_printButton;
VkMenuItem *_separator1;
VkMenuItem *_closeButton;
VkMenuItem *_exitButton;
VkSubMenu *_editPane;
VkMenuItem *_theVkUndoManagerButton;
VkMenuItem *_cutButton;
VkMenuItem *_copyButton;
VkMenuItem *_pasteButton;
VkSubMenu *_viewPane;
VkMenuItem *_imgInfo;
VkMenuItem *_mraInfo;
VkMenuItem *_pcmraLocalizer;
VkSubMenu *_user;
VkMenuItem *_novies;
VkMenuItem *_expert;

```

```
// Member functions called from callbacks
```

```

virtual void close ( Widget, XtPointer );
virtual void copy ( Widget, XtPointer );
virtual void cut ( Widget, XtPointer );
virtual void expertCallback ( Widget, XtPointer );
virtual void imgInfoCallback ( Widget, XtPointer );
virtual void mraInfoCallback ( Widget, XtPointer );
virtual void newFile ( Widget, XtPointer );
virtual void noviesCallback ( Widget, XtPointer );
virtual void openFile ( Widget, XtPointer );
virtual void paste ( Widget, XtPointer );
virtual void pcmraCutCallback ( Widget, XtPointer );
virtual void print ( Widget, XtPointer );
virtual void quit ( Widget, XtPointer );
virtual void save ( Widget, XtPointer );
virtual void saveas ( Widget, XtPointer );

```

```
//---- Start editable code block: VkwindowMainWindow protected
```

```
//---- End editable code block: VkwindowMainWindow protected
```

```
private:
```

```
// Callbacks to interface with Motif
```

```

static void closeCallback ( Widget, XtPointer, XtPointer );
static void copyCallback ( Widget, XtPointer, XtPointer );
static void cutCallback ( Widget, XtPointer, XtPointer );
static void expertCallbackCallback ( Widget, XtPointer, XtPointer );
static void imgInfoCallbackCallback ( Widget, XtPointer, XtPointer );
static void mraInfoCallbackCallback ( Widget, XtPointer, XtPointer );
static void newFileCallback ( Widget, XtPointer, XtPointer );
static void noviesCallbackCallback ( Widget, XtPointer, XtPointer );
static void openFileCallback ( Widget, XtPointer, XtPointer );
static void pasteCallback ( Widget, XtPointer, XtPointer );
static void pcmraCutCallbackCallback ( Widget, XtPointer, XtPointer );
static void printCallback ( Widget, XtPointer, XtPointer );
static void quitCallback ( Widget, XtPointer, XtPointer );
static void saveCallback ( Widget, XtPointer, XtPointer );
static void saveasCallback ( Widget, XtPointer, XtPointer );

```

```
static String _defaultVkwindowMainWindowResources[];
```

```
//---- Start editable code block: VkwindowMainWindow private
```

```
//---- End editable code block: VkwindowMainWindow private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```
#ifndef MESSAGESRIGHT_H
#define MESSAGESRIGHT_H
```

752

```
#include "Flow.h"
#include "Points.h"
```

```
typedef struct {
```

```
    Boolean  show_status;
```

```
    int      img_select;  //ROI, Whole, Reference, Other
```

```
    int      img_type;    //CT, MR, PCMR
```

```
    int      img_pcmra_type;  //MAGNITUDE, PHASE, VELOCITY
```

```
    int      img_anatomy;   //HEAD, NECK, BONE
```

```
    int      img_number;
```

```
    int      img_number_prev;
```

```
    float    img_zoom;
```

```
    int      img_scale_type;
```

```
    float    img_winCenter;
```

```
    float    img_winWidth;
```

```
    float    low, high;
```

```
    float    lowGrayWhole, highGrayWhole;
```

```
    float    lowGrayROI, highGrayROI;
```

```
    float    lowGrayRef, highGrayRef;
```

```
    float    lowGrayOther, highGrayOther;
```

```
    float    lowColorWhole, highColorWhole;
```

```
    float    lowColorROI, highColorROI;
```

```
    float    lowColorRef, highColorRef;
```

```
    float    lowColorOther, highColorOther;
```

```
    float    lowMagGrayWhole, highMagGrayWhole;
```

```
    float    lowMagGrayROI, highMagGrayROI;
```

```
    float    lowMagGrayRef, highMagGrayRef;
```

```
    float    lowMagGrayOther, highMagGrayOther;
```

```
    float    lowMagColorWhole, highMagColorWhole;
```

```
    float    lowMagColorROI, highMagColorROI;
```

```
    float    lowMagColorRef, highMagColorRef;
```

```
    float    lowMagColorOther, highMagColorOther;
```

```
    float    lowPhaGrayWhole, highPhaGrayWhole;
```

```
    float    lowPhaGrayROI, highPhaGrayROI;
```

```
    float    lowPhaGrayRef, highPhaGrayRef;
```

```
    float    lowPhaGrayOther, highPhaGrayOther;
```

```
    float    lowPhaColorWhole, highPhaColorWhole;
```

```
    float    lowPhaColorROI, highPhaColorROI;
```

```
    float    lowPhaColorRef, highPhaColorRef;
```

```
    float    lowPhaColorOther, highPhaColorOther;
```

```
    int      img_visual_type;  //GRAY, COLOR
```

```
    int      img_space;  //2D, 3D
```

```
    int      histo_status;
```

```
    float    histo_min;
```

```
    float    histo_max;
```

```
    int      roi_type;
```

```
    int      roi_action;
```

```
    int      roi_x, roi_y, roi_w, roi_h;
```

```
int      num_imgs;
int      num_cardiacs;

char     vesselName[100];
char     userName[100];
int      flowDir;
int      flowDir2;

Boolean  show_detail;

int      animate_mode;

int      velocity_select;
float    velocity_ratio;
int      low_magthresh;

int      flow_select;
int      flow_method;
FlowPara *flows;
int      flow_noiseLevel;

float    ratio3D;
int      camera;
float    YPos3D;
float    Height3D;
int      Fixed3D;
int      flow3D;

int      publish;
int      flow3DDir;
float    HR;

int      roi_changed;
unsigned char **roi_mask;
unsigned char **roi_flow;
unsigned char **roi_back;

Points   *roi_points;

char     pubDir[300];

} MessagesRight;

#endif
```

```
#ifndef POINT_H  
#define POINT_H
```

754

```
typedef struct {  
    float  x;  
    float  y;  
} Point;
```

```
#endif
```

```

virtual void anatomy ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void patients ( Widget, XtPointer );
//----- Start editable code block: BbLConfig protected
//----- End editable code block: BbLConfig protected
private:
// Array of default resources
static String _defaultBbLConfigUIResources[];
// Callbacks to interface with Motif
static void anatomyCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );
static void patientsCallback ( Widget, XtPointer, XtPointer );
//----- Start editable code block: BbLConfig private
//----- End editable code block: BbLConfig private
----- Start editable code block: End of generated code
--- End editable code block: End of generated code
if

```



```

////////////////////////////////////
//
// Header file for BbLPCMRA
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLPCMRAUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBLPCMRA_H
#define BBLPCMRA_H
#include "BbLPCMRAUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbLPCMRA class declaration

class BbLPCMRA : public BbLPCMRAUI
{
public:
    BbLPCMRA ( const char *, Widget );
    BbLPCMRA ( const char * );
    ~BbLPCMRA();
    const char * className();

    static VkComponent *CreateBbLPCMRA( const char *name, Widget parent );

//---- Start editable code block: BbLPCMRA public
    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

//---- End editable code block: BbLPCMRA public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbLPCMRAUI

    virtual void doButtonHideLocalizer ( Widget, XtPointer );
    virtual void doButtonShowLocalizer ( Widget, XtPointer );

//---- Start editable code block: BbLPCMRA protected

```

//---- End editable code block: BbLPCMRA protected

702

private:

static void* RegisterBbLPCMRAInterface();

//---- Start editable code block: BbLPCMRA private

//---- End editable code block: BbLPCMRA private

};

//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif

```

////////////////////////////////////
//
// Header file for BbLPCMRAUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLPCMRA is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLPCMRA files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBLPCMRAUI_H
#define BBLPCMRAUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class BbLPCMRAUI : public VkComponent
{
public:
    BbLPCMRAUI ( const char *, Widget );
    BbLPCMRAUI ( const char * );
    ~BbLPCMRAUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbLPCMRA public

    /*--- End editable code block: BbLPCMRA public

protected:

    // Widgets created by this class

    Widget _bbLPCMRA;
    Widget _buttonHideLocalizer;
    Widget _buttonShowLocalizer;

    // These virtual functions are called from the private callbacks (below)
    // Intended to be overridden in derived classes to define actions

```

```

virtual void doButtonHideLocalizer ( Widget, XtPointer );
virtual void doButtonShowLocalizer ( Widget, XtPointer );

//----- Start editable code block: BbLPCMRA protected

//----- End editable code block: BbLPCMRA protected

private:

// Array of default resources

static String      _defaultBbLPCMRAUIResources[];

// Callbacks to interface with Motif

static void doButtonHideLocalizerCallback ( Widget, XtPointer, XtPointer );
static void doButtonShowLocalizerCallback ( Widget, XtPointer, XtPointer );

//----- Start editable code block: BbLPCMRA private

//----- End editable code block: BbLPCMRA private
};
//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

#endif

```

User: meide
Host: phoenix
Class: phoenix
Job: BbHistogram.h

```

////////////////////////////////////
//
// Header file for BbLROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBLROI_H
#define BBLROI_H
#include "BbLROIUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbLROI class declaration
class BbLROI : public BbLROIUI
{
public:

    BbLROI ( const char *, Widget );
    BbLROI ( const char * );
    ~BbLROI();
    const char * className();

    static VkComponent *CreateBbLROI( const char *name, Widget parent );

    //---- Start editable code block: BbLROI public
    int _roi_mode;
    int _roi_type;
    int _roi_action;
    int _roi_color;

    void init();
    void initROI();
    void init2();
    void changeROI();
    void set_color();
    void set_type();

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    //---- End editable code block: BbLROI public

```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbLROIUI
```

```
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doButtonHide ( Widget, XtPointer );
virtual void doButtonShow ( Widget, XtPointer );
virtual void doOptionBlack ( Widget, XtPointer );
virtual void doOptionBlue ( Widget, XtPointer );
virtual void doOptionDraw ( Widget, XtPointer );
virtual void doOptionEllipse ( Widget, XtPointer );
virtual void doOptionEraseLeft ( Widget, XtPointer );
virtual void doOptionEraseRight ( Widget, XtPointer );
virtual void doOptionFreeHand ( Widget, XtPointer );
virtual void doOptionGreen ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionPolygon ( Widget, XtPointer );
virtual void doOptionRectangle ( Widget, XtPointer );
virtual void doOptionRed ( Widget, XtPointer );
virtual void doOptionWhite ( Widget, XtPointer );
virtual void doOptionYellow ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLROI protected
```

```
//---- End editable code block: BbLROI protected
```

private:

```
static void* RegisterBbLROIInterface();
```

```
//---- Start editable code block: BbLROI private
```

```
//---- End editable code block: BbLROI private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbLROIUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLROI is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLROI files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//

```

```

////////////////////////////////////
#ifndef BBLROIUI_H
#define BBLROIUI_H
#include <Vk/VkComponent.h>

```

```

//---- Start editable code block: headers and declarations

```

```

//---- End editable code block: headers and declarations

```

```

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class VkMenuItem;
class BbLROIUI : public VkComponent
{

```

```

    public:

```

```

        BbLROIUI ( const char *, Widget );
        BbLROIUI ( const char * );
        ~BbLROIUI();
        void create ( Widget );
        const char * className();

```

```

        //---- Start editable code block: BbLROI public

```

```

        //---- End editable code block: BbLROI public

```

```

    protected:

```

```

        // Widgets created by this class

```

```

        Widget _bbLROI;
        Widget _buttonAcceptArea;
        Widget _buttonHide;
        Widget _buttonShow;

```



```
VkOptionMenu *_optionMenu6;  
VkOptionMenu *_optionMenuColor;  
VkOptionMenu *_optionMenuROIType;
```

```
VkMenuItem *_optionBlack;  
VkMenuItem *_optionBlue;  
VkMenuItem *_optionDraw;  
VkMenuItem *_optionEllipse;  
VkMenuItem *_optionEraseLeft;  
VkMenuItem *_optionEraseRight;  
VkMenuItem *_optionFreeHand;  
VkMenuItem *_optionGreen;  
VkMenuItem *_optionModify;  
VkMenuItem *_optionPolygon;  
VkMenuItem *_optionRectangle;  
VkMenuItem *_optionRed;  
VkMenuItem *_optionWhite;  
VkMenuItem *_optionYellow;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doButtonAccept ( Widget, XtPointer );  
virtual void doButtonHide ( Widget, XtPointer );  
virtual void doButtonShow ( Widget, XtPointer );  
virtual void doOptionBlack ( Widget, XtPointer );  
virtual void doOptionBlue ( Widget, XtPointer );  
virtual void doOptionDraw ( Widget, XtPointer );  
virtual void doOptionEllipse ( Widget, XtPointer );  
virtual void doOptionEraseLeft ( Widget, XtPointer );  
virtual void doOptionEraseRight ( Widget, XtPointer );  
virtual void doOptionFreeHand ( Widget, XtPointer );  
virtual void doOptionGreen ( Widget, XtPointer );  
virtual void doOptionModify ( Widget, XtPointer );  
virtual void doOptionPolygon ( Widget, XtPointer );  
virtual void doOptionRectangle ( Widget, XtPointer );  
virtual void doOptionRed ( Widget, XtPointer );  
virtual void doOptionWhite ( Widget, XtPointer );  
virtual void doOptionYellow ( Widget, XtPointer );
```

```
//----- Start editable code block: BbLROI protected
```

```
//----- End editable code block: BbLROI protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbLROIUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );  
static void doButtonHideCallback ( Widget, XtPointer, XtPointer );  
static void doButtonShowCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBlackCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBlueCallback ( Widget, XtPointer, XtPointer );  
static void doOptionDrawCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEllipseCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEraseLeftCallback ( Widget, XtPointer, XtPointer );  
static void doOptionEraseRightCallback ( Widget, XtPointer, XtPointer );
```

```
static void doOptionFreeHandCallback ( Widget, XtPointer, XtPointer );
static void doOptionGrCallback ( Widget, XtPointer, XtPointer );
static void doOptionMoCallback ( Widget, XtPointer, XtPointer );
static void doOptionPolygonCallback ( Widget, XtPointer, XtPointer );
static void doOptionRectangleCallback ( Widget, XtPointer, XtPointer );
static void doOptionRedCallback ( Widget, XtPointer, XtPointer );
static void doOptionWhiteCallback ( Widget, XtPointer, XtPointer );
static void doOptionYellowCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbLROI private
```

```
//---- End editable code block: BbLROI private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

/////////////////////////////////////////////////////////////////
//
// Header file for BbLWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbLWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
/////////////////////////////////////////////////////////////////
#ifndef BBLWAVEFORM_H
#define BBLWAVEFORM_H
#include "BbLWaveformUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbLWaveform class declaration

class BbLWaveform : public BbLWaveformUI
{
public:
    BbLWaveform ( const char *, Widget );
    BbLWaveform ( const char * );
    ~BbLWaveform();
    const char * className();

    static VkComponent *CreateBbLWaveform( const char *name, Widget parent );

//---- Start editable code block: BbLWaveform public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel); _current_vessel = (

    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);

//---- End editable code block: BbLWaveform public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in BbLWaveformUI
```

704

```
virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessellWaveform ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLWaveform protected
```

```
//---- End editable code block: BbLWaveform protected
```

```
private:
```

```
static void* RegisterBbLWaveformInterface();
```

```
//---- Start editable code block: BbLWaveform private
```

```
//---- End editable code block: BbLWaveform private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbLWaveformUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbLWaveform is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbLWaveform files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBLWAVEFORMUI_H
#define BBLWAVEFORMUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuToggle;
class VkMenuItem;
class BbLWaveformUI : public VkComponent
{
public:
    BbLWaveformUI ( const char *, Widget );
    BbLWaveformUI ( const char * );
    ~BbLWaveformUI();
    void create ( Widget );
    const char * className();

/*--- Start editable code block: BbLWaveform public

/*--- End editable code block: BbLWaveform public

protected:

    // Widgets created by this class

    Widget _bbLWaveform;
    Widget _labelCurrentNum1;
    Widget _labelMaxNum2;
    Widget _labelMinNum2;

```

```
Widget _labelUnit1;  
Widget _scrolledListUnit1;  
Widget _scrolledWindow1;
```

706

```
VkOptionMenu *_optionMenuFlow;
```

```
VkMenuItem *_optionASV;  
VkMenuItem *_optionArea;  
VkMenuItem *_optionBSV;  
VkMenuItem *_optionPSV;  
VkMenuItem *_optionVFR;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doOptionASV ( Widget, XtPointer );  
virtual void doOptionArea ( Widget, XtPointer );  
virtual void doOptionBSV ( Widget, XtPointer );  
virtual void doOptionPSV ( Widget, XtPointer );  
virtual void doOptionVFR ( Widget, XtPointer );  
virtual void vessellWaveform ( Widget, XtPointer );
```

```
//---- Start editable code block: BbLWaveform protected
```

```
//---- End editable code block: BbLWaveform protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbLWaveformUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doOptionASVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );  
static void vessellWaveformCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbLWaveform private
```

```
//---- End editable code block: BbLWaveform private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbLROI.h

707

```

////////////////////////////////////
//
// Header file for BbRHistogram
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRHistogramUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRHISTOGRAM_H
#define BBRHISTOGRAM_H
#include "BbRHistogramUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbRHistogram class declaration

class BbRHistogram : public BbRHistogramUI
{
public:
    BbRHistogram ( const char *, Widget );
    BbRHistogram ( const char * );
    ~BbRHistogram();
    const char * className();

    static VkComponent *CreateBbRHistogram( const char *name, Widget parent );

//---- Start editable code block: BbRHistogram public

    int _winWidth;
    int _winCenter;

    void init();

    void update(float, float);
    void update_lowhigh(float, float);

    void update_width(int);
    void update_center(int);

    void set_mapLabels();

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

```


protected:

```
// These functions will be called as a result of callbacks
// registered in BbRHistogramUI
```

```
virtual void AneurysmFlow ( Widget, XtPointer );
virtual void BlackFlow ( Widget, XtPointer );
virtual void WhiteFlow ( Widget, XtPointer );
virtual void abdomen2 ( Widget, XtPointer );
virtual void bone2 ( Widget, XtPointer );
virtual void centerDrag2 ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head2 ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung2 ( Widget, XtPointer );
virtual void mediastinum2 ( Widget, XtPointer );
virtual void spine2 ( Widget, XtPointer );
virtual void widthDrag2 ( Widget, XtPointer );
```

//---- Start editable code block: BbRHistogram protected

//---- End editable code block: BbRHistogram protected

private:

static void* RegisterBbRHistogramInterface();

//---- Start editable code block: BbRHistogram private

//---- End editable code block: BbRHistogram private

```
};
//---- Start editable code block: End of generated code
```

//---- End editable code block: End of generated code

#endif

```

////////////////////////////////////
//
// Header file for BbRHistogramUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRHistogram is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRHistogram files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRHISTOGRAMUI_H
#define BBRHISTOGRAMUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuToggle;
class VkMenuAction;
class BbRHistogramUI : public VkComponent
{
public:
    BbRHistogramUI ( const char *, Widget );
    BbRHistogramUI ( const char * );
    ~BbRHistogramUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbRHistogram public

    void create_mapLabels();
    void remove_mapLabels();

    Widget _bbRHistogram;
    Widget _dialCenter2;
    Widget _dialWidth2;
    Widget _labelLHistoHigh2;
    Widget _labelLHistoLow2;
    Widget _labelLHistoMax2;
    Widget _labelLHistoMin2;
    Widget _labelMap1;

```

```

Widget  _labelMap2;
Widget  _labelMap3;
Widget  _labelMap4;
Widget  _labelMap5;
Widget  _labelMap6;
Widget  _labelMap7;
Widget  _labelMap8;
Widget  _labelMap9;

```

```
//---- End editable code block: BbRHistogram public
```

```
protected:
```

```
// Widgets created by this class
```

```

VkOptionMenu  *_optionMenuLHistogram21;
VkOptionMenu  *_optionMenuRhyst;

```

```

VkMenuItem  *_optionAbdomen1;
VkMenuItem  *_optionAneurysmFlow;
VkMenuItem  *_optionBlackFlow;
VkMenuItem  *_optionBone1;
VkMenuItem  *_optionCoarse;
VkMenuItem  *_optionFine;
VkMenuItem  *_optionHead1;
VkMenuItem  *_optionLung1;
VkMenuItem  *_optionMapping;
VkMenuItem  *_optionMediaStinum1;
VkMenuItem  *_optionROI1;
VkMenuItem  *_optionSpine1;
VkMenuItem  *_optionUpdate;
VkMenuItem  *_optionWhiteFlow;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void AneurysmFlow ( Widget, XtPointer );
virtual void BlackFlow ( Widget, XtPointer );
virtual void WhiteFlow ( Widget, XtPointer );
virtual void abdomen2 ( Widget, XtPointer );
virtual void bone2 ( Widget, XtPointer );
virtual void centerDrag2 ( Widget, XtPointer );
virtual void doOptionCoarse ( Widget, XtPointer );
virtual void doOptionFine ( Widget, XtPointer );
virtual void doOptionMapping ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionUpdate ( Widget, XtPointer );
virtual void head2 ( Widget, XtPointer );
virtual void highChg ( Widget, XtPointer );
virtual void lowChg ( Widget, XtPointer );
virtual void lung2 ( Widget, XtPointer );
virtual void mediastinum2 ( Widget, XtPointer );
virtual void spine2 ( Widget, XtPointer );
virtual void widthDrag2 ( Widget, XtPointer );

```

```
//---- Start editable code block: BbRHistogram protected
```

```
//---- End editable code block: BbRHistogram protected
```

private:

712

```
// Array of default resources

static String      _defaultBbRHistogramUIResources[];

// Callbacks to interface with Motif

static void AneurysmFlowCallback ( Widget, XtPointer, XtPointer );
static void BlackFlowCallback ( Widget, XtPointer, XtPointer );
static void WhiteFlowCallback ( Widget, XtPointer, XtPointer );
static void abdomen2Callback ( Widget, XtPointer, XtPointer );
static void bone2Callback ( Widget, XtPointer, XtPointer );
static void centerDrag2Callback ( Widget, XtPointer, XtPointer );
static void doOptionCoarseCallback ( Widget, XtPointer, XtPointer );
static void doOptionFineCallback ( Widget, XtPointer, XtPointer );
static void doOptionMappingCallback ( Widget, XtPointer, XtPointer );
static void doOptionROICallback ( Widget, XtPointer, XtPointer );
static void doOptionUpdateCallback ( Widget, XtPointer, XtPointer );
static void head2Callback ( Widget, XtPointer, XtPointer );
static void highChgCallback ( Widget, XtPointer, XtPointer );
static void lowChgCallback ( Widget, XtPointer, XtPointer );
static void lung2Callback ( Widget, XtPointer, XtPointer );
static void mediastinum2Callback ( Widget, XtPointer, XtPointer );
static void spine2Callback ( Widget, XtPointer, XtPointer );
static void widthDrag2Callback ( Widget, XtPointer, XtPointer );

//---- Start editable code block: BbRHistogram private

//---- End editable code block: BbRHistogram private
};
//---- Start editable code block: End of generated code

//---- End editable code block: End of generated code

#endif
```

```

////////////////////////////////////
//
// Header file for BbRROI
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRROIUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRROI_H
#define BBRROI_H
#include "BbRROIUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbRROI class declaration

class BbRROI : public BbRROIUI
{
public:

    BbRROI ( const char *, Widget );
    BbRROI ( const char * );
    ~BbRROI();
    const char * className();

    static VkComponent *CreateBbRROI( const char *name, Widget parent );

//---- Start editable code block: BbRROI public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _ROI_NO;
    void draw_ROI();
    void draw_AllROI(int);
    void show_current(int i);
    void show_total(int i);
    void modify();
    void add_AllROI(int, int);

    int modify(char *name);

    int _mode;
    int _roi_nn;
    int _frame;

```

```
void set_list();
void draw_ROINeighbor()
```

714

```
//----- End editable code block: BbRROI public
```

protected:

```
// These functions will be called as a result of callbacks
// registered in BbRROIUI
```

```
virtual void NextNeighbor ( Widget, XtPointer );
virtual void PrevROI ( Widget, XtPointer );
virtual void ROIName ( Widget, XtPointer );
virtual void doButtonAcceptROI ( Widget, XtPointer );
virtual void doButtonRemove ( Widget, XtPointer );
virtual void doButtonSaveROI ( Widget, XtPointer );
virtual void doOptionBackFlow ( Widget, XtPointer );
virtual void doOptionHide ( Widget, XtPointer );
virtual void doOptionHideNeighbor ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionModify3D ( Widget, XtPointer );
virtual void doOptionOpenROI ( Widget, XtPointer );
virtual void doOptionROIFlow ( Widget, XtPointer );
virtual void doOptionSave3D ( Widget, XtPointer );
virtual void doOptionShow ( Widget, XtPointer );
virtual void doOptionShow3D ( Widget, XtPointer );
virtual void doOptionShowAll ( Widget, XtPointer );
virtual void doOptionShowAllNeighbor ( Widget, XtPointer );
virtual void doOptionShowNeighbor ( Widget, XtPointer );
virtual void rois ( Widget, XtPointer );
```

```
//----- Start editable code block: BbRROI protected
```

```
//----- End editable code block: BbRROI protected
```

private:

```
static void* RegisterBbRROIInterface();
```

```
//----- Start editable code block: BbRROI private
```

```
//----- End editable code block: BbRROI private
```

```
};
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbRHistogram.h

715

```

////////////////////////////////////
//
// Header file for BbRROIUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRROI is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRROI files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRROIUI_H
#define BBRROIUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbRROIUI : public VkComponent
{
public:

    BbRROIUI ( const char *, Widget );
    BbRROIUI ( const char * );
    ~BbRROIUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbRROI public

    //--- End editable code block: BbRROI public

protected:

    // Widgets created by this class

    Widget _arrow1;
    Widget _arrowPrevROI;
    Widget _bbRROI;
    Widget _buttonAcceptROI;

```



```

Widget _buttonRemoveROI;
Widget _buttonSaveROI;
Widget _labelNumCurr;
Widget _labelNumROI;
Widget _labelROIName;
Widget _scrolledListROIS;
Widget _scrolledWindow6;
Widget _separator3;
Widget _textfieldROIName;

```

```

VkOptionMenu *_optionMenu5;
VkOptionMenu *_optionMenu7;
VkOptionMenu *_optionMenu8;

```

```

VkMenuItem *_optionBackFlow;
VkMenuItem *_optionHide;
VkMenuItem *_optionHideNeighbor;
VkMenuItem *_optionModify3D;
VkMenuItem *_optionModifyROI;
VkMenuItem *_optionOpenROI;
VkMenuItem *_optionROIFlow;
VkMenuItem *_optionSave3D;
VkMenuItem *_optionShow;
VkMenuItem *_optionShow3D;
VkMenuItem *_optionShowAll;
VkMenuItem *_optionShowAllNeighbor;
VkMenuItem *_optionShowNeighbor;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void NextNeighbor ( Widget, XtPointer );
virtual void PrevROI ( Widget, XtPointer );
virtual void ROIName ( Widget, XtPointer );
virtual void doButtonAcceptROI ( Widget, XtPointer );
virtual void doButtonRemove ( Widget, XtPointer );
virtual void doButtonSaveROI ( Widget, XtPointer );
virtual void doOptionBackFlow ( Widget, XtPointer );
virtual void doOptionHide ( Widget, XtPointer );
virtual void doOptionHideNeighbor ( Widget, XtPointer );
virtual void doOptionModify ( Widget, XtPointer );
virtual void doOptionModify3D ( Widget, XtPointer );
virtual void doOptionOpenROI ( Widget, XtPointer );
virtual void doOptionROIFlow ( Widget, XtPointer );
virtual void doOptionSave3D ( Widget, XtPointer );
virtual void doOptionShow ( Widget, XtPointer );
virtual void doOptionShow3D ( Widget, XtPointer );
virtual void doOptionShowAll ( Widget, XtPointer );
virtual void doOptionShowAllNeighbor ( Widget, XtPointer );
virtual void doOptionShowNeighbor ( Widget, XtPointer );
virtual void rois ( Widget, XtPointer );

```

```

//---- Start editable code block: BbRROI protected

```

```

//---- End editable code block: BbRROI protected

```

```

private:

```

```

// Array of default resources

```

```

static String _defaultBbRROIUIResources[];

```

```
static void NextNeighborCallback ( Widget, XtPointer, XtPointer );
static void PrevROIDCallback ( Widget, XtPointer, XtPointer );
static void ROINameCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptROIDCallback ( Widget, XtPointer, XtPointer );
static void doButtonRemoveCallback ( Widget, XtPointer, XtPointer );
static void doButtonSaveROIDCallback ( Widget, XtPointer, XtPointer );
static void doOptionBackFlowCallback ( Widget, XtPointer, XtPointer );
static void doOptionHideCallback ( Widget, XtPointer, XtPointer );
static void doOptionHideNeighborCallback ( Widget, XtPointer, XtPointer );
static void doOptionModifyCallback ( Widget, XtPointer, XtPointer );
static void doOptionModify3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionOpenROIDCallback ( Widget, XtPointer, XtPointer );
static void doOptionROIFlowCallback ( Widget, XtPointer, XtPointer );
static void doOptionSave3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowCallback ( Widget, XtPointer, XtPointer );
static void doOptionShow3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowAllCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowAllNeighborCallback ( Widget, XtPointer, XtPointer );
static void doOptionShowNeighborCallback ( Widget, XtPointer, XtPointer );
static void roisCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbRROI private
```

```
//---- End editable code block: BbRROI private
```

```
};
```

```
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRTable
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRTableUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
-// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRTABLE_H
#define BBRTABLE_H
#include "BbRTableUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbRTable class declaration

class BbRTable : public BbRTableUI
{
public:

    BbRTable ( const char *, Widget );
    BbRTable ( const char * );
    ~BbRTable();
    const char * className();

    static VkComponent *CreateBbRTable( const char *name, Widget parent );

//---- Start editable code block: BbRTable public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel3); _current_vessel =

    void show_info();
    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);
    void set_list(int num, float *x);

//---- End editable code block: BbRTable public

protected:

```

```
// These functions will be called as a result of callbacks
// registered in BbRTableUI

virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessel ( Widget, XtPointer );
virtual void vesselRTable ( Widget, XtPointer );

//----- Start editable code block: BbRTable protected

//----- End editable code block: BbRTable protected

private:

static void* RegisterBbRTableInterface();

//----- Start editable code block: BbRTable private

//----- End editable code block: BbRTable private

};
//----- Start editable code block: End of generated code

//----- End editable code block: End of generated code

#endif
```

```

////////////////////////////////////
//
// Header file for BbRTableUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRTable is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbRTable files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRTABLEUI_H
#define BBRTABLEUI_H
#include <Vk/VkComponent.h>

//--- Start editable code block: headers and declarations

//--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbRTableUI : public VkComponent
{
public:

    BbRTableUI ( const char *, Widget );
    BbRTableUI ( const char * );
    ~BbRTableUI();
    void create ( Widget );
    const char * className();

    //--- Start editable code block: BbRTable public

    //--- End editable code block: BbRTable public

protected:

    // Widgets created by this class

    Widget _bbRTable;
    Widget _labelAverage1;
    Widget _labelAverageNum1;
    Widget _labelHeartRate1;

```

```

Widget  _labelHeartRateNum1;
Widget  _labelMax1;
Widget  _labelMaxNum1;
Widget  _labelMin1;
Widget  _labelMinNum1;
Widget  _labelUnits;
Widget  _labelUnitsNum;
Widget  _labelVolume1;
Widget  _labelVolumeNum1;
Widget  _scrolledListVessel1;
Widget  _scrolledListVessel3;
Widget  _scrolledWindow2;
Widget  _scrolledWindow4;

```

```
VkOptionMenu *_optionMenuFlow2;
```

```

VkMenuItem *_optionASV2;
VkMenuItem *_optionArea2;
VkMenuItem *_optionBSV2;
VkMenuItem *_optionPSV2;
VkMenuItem *_optionVFR2;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void doOptionASV ( Widget, XtPointer );
virtual void doOptionArea ( Widget, XtPointer );
virtual void doOptionBSV ( Widget, XtPointer );
virtual void doOptionPSV ( Widget, XtPointer );
virtual void doOptionVFR ( Widget, XtPointer );
virtual void vessel ( Widget, XtPointer );
virtual void vesselRTable ( Widget, XtPointer );

```

```
//---- Start editable code block: BbRTable protected
```

```
//---- End editable code block: BbRTable protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbRTableUIResources[];
```

```
// Callbacks to interface with Motif
```

```

static void doOptionASVCallback ( Widget, XtPointer, XtPointer );
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );
static void vesselCallback ( Widget, XtPointer, XtPointer );
static void vesselRTableCallback ( Widget, XtPointer, XtPointer );

```

```
//---- Start editable code block: BbRTable private
```

```
//---- End editable code block: BbRTable private
```

```

};
//---- Start editable code block: End of generated code

```

```
//---- End editable code block: End of generated code
```

#endif

723

```

////////////////////////////////////
//
// Header file for BbRWaveform
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbRWaveformUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBRWAVEFORM_H
#define BBRWAVEFORM_H
#include "BbRWaveformUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbRWaveform class declaration

class BbRWaveform : public BbRWaveformUI
{
public:

    BbRWaveform ( const char *, Widget );
    BbRWaveform ( const char * );
    ~BbRWaveform();
    const char *  className();

    static VkComponent *CreateBbRWaveform( const char *name, Widget parent );

//---- Start editable code block: BbRWaveform public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    int _current_vessel;
    void add_vessel(char *str);
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel2); _current_vessel =

    void set_unit(char *str);
    void set_info(float minI, float maxI, float avg);

//---- End editable code block: BbRWaveform public

protected:

```



```
// These functions will be called as a result of calls  
// registered in BbRWaveformUI
```

725

```
virtual void doOptionASV ( Widget, XtPointer );  
virtual void doOptionArea ( Widget, XtPointer );  
virtual void doOptionBSV ( Widget, XtPointer );  
virtual void doOptionPSV ( Widget, XtPointer );  
virtual void doOptionVFR ( Widget, XtPointer );  
virtual void vesselRWaveform ( Widget, XtPointer );
```

```
//----- Start editable code block: BbRWaveform protected
```

```
//----- End editable code block: BbRWaveform protected
```

```
private:
```

```
static void* RegisterBbRWaveformInterface();
```

```
//----- Start editable code block: BbRWaveform private
```

```
//----- End editable code block: BbRWaveform private
```

```
};  
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbRWaveformUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbRWaveform is instantiated
//
// To extend or alter the behavior of this class, you should
-// modify the BbRWaveform files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBRWAVEFORMUI_H
#define BBRWAVEFORMUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItemToggle;
class VkMenuItem;
class BbRWaveformUI : public VkComponent
{
public:
    BbRWaveformUI ( const char *, Widget );
    BbRWaveformUI ( const char * );
    ~BbRWaveformUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbRWaveform public

    //---- End editable code block: BbRWaveform public

protected:

    // Widgets created by this class

    Widget _bbRWaveform;
    Widget _labelCurrentNum;
    Widget _labelMaxNum;
    Widget _labelMinNum;

```

```
Widget _labelUnit;  
Widget _scrolledListVessel2;  
Widget _scrolledWindows;
```

727

```
VkOptionMenu *_optionMenuFlow1;
```

```
VkMenuItem *_optionASV1;  
VkMenuItem *_optionArea1;  
VkMenuItem *_optionBSV1;  
VkMenuItem *_optionPSV1;  
VkMenuItem *_optionVFR1;
```

```
// These virtual functions are called from the private callbacks (below)  
// Intended to be overridden in derived classes to define actions
```

```
virtual void doOptionASV ( Widget, XtPointer );  
virtual void doOptionArea ( Widget, XtPointer );  
virtual void doOptionBSV ( Widget, XtPointer );  
virtual void doOptionPSV ( Widget, XtPointer );  
virtual void doOptionVFR ( Widget, XtPointer );  
virtual void vesselRWaveform ( Widget, XtPointer );
```

```
//----- Start editable code block: BbRWaveform protected
```

```
//----- End editable code block: BbRWaveform protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbRWaveformUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void doOptionASVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionAreaCallback ( Widget, XtPointer, XtPointer );  
static void doOptionBSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionPSVCallback ( Widget, XtPointer, XtPointer );  
static void doOptionVFRCallback ( Widget, XtPointer, XtPointer );  
static void vesselRWaveformCallback ( Widget, XtPointer, XtPointer );
```

```
//----- Start editable code block: BbRWaveform private
```

```
//----- End editable code block: BbRWaveform private
```

```
};  
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, Bb is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the Bb files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBUI_H
#define BBUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include "Flow.h"

/*--- End editable code block: headers and declarations

// Externally defined classes referenced by this class:

class DeckLTabbedDeck;
class DeckRTabbedDeck;

class VkOptionMenu;
class VkMenuItem;
class VkMenuItemToggle;
class VkMenuItem;
class BbUI : public VkComponent
{
public:
    BbUI ( const char *, Widget );
    BbUI ( const char * );
    ~BbUI();
    void create ( Widget );
    const char * className();

/*--- Start editable code block: Bb public

ObjectManager *_objMag;
void set(class ObjectManager *);

void init();

```

```

void init_patient();

Flow * init_flow(int num);
void remove_flow();

Widget _labelImgNumber;

//---- End editable code block: Bb public

```

protected:

```

// Classes created by this class

class DeckRTabbedDeck *_deckR;
class DeckLTabbedDeck *_deckL;

// Widgets created by this class

Widget _arrowNext;
Widget _arrowPrev;
Widget _bb;
//Widget _labelImgNumber;
Widget _separatorBottom;
Widget _separatorMiddle;
Widget _separatorTop;

VkOptionMenu *_optionMenuAnimate;
VkOptionMenu *_optionMenuPCMR;
VkOptionMenu *_optionMenuSelect;
VkOptionMenu *_optionMenuSpace;
VkOptionMenu *_optionMenuVisual;

VkMenuItem *_option3D;
VkMenuItem *_optionAnimate;
VkMenuItem *_optionColor2D;
VkMenuItem *_optionGray2D;
VkMenuItem *_optionMagnitude;
VkMenuItem *_optionNewAnimate;
VkMenuItem *_optionOther;
VkMenuItem *_optionPhase;
VkMenuItem *_optionROI;
VkMenuItem *_optionReference;
VkMenuItem *_optionSimple;
VkMenuItem *_optionSpline;
VkMenuItem *_optionStopAnimate;
VkMenuItem *_optionVelocity;
VkMenuItem *_optionWhole;

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

virtual void Next ( Widget, XtPointer );
virtual void Prev ( Widget, XtPointer );
virtual void doOption3D ( Widget, XtPointer );
virtual void doOptionAnimate ( Widget, XtPointer );
virtual void doOptionColor2D ( Widget, XtPointer );
virtual void doOptionGray2D ( Widget, XtPointer );
virtual void doOptionMagnitude ( Widget, XtPointer );
virtual void doOptionNewAnimate ( Widget, XtPointer );
virtual void doOptionOther ( Widget, XtPointer );
virtual void doOptionPhase ( Widget, XtPointer );
virtual void doOptionROI ( Widget, XtPointer );
virtual void doOptionReference ( Widget, XtPointer );

```

```

virtual void doOptionSimple ( Widget, XtPointer );
virtual void doOptionSpline ( Widget, XtPointer );
virtual void doOptionStopAnimate ( Widget, XtPointer );
virtual void doOptionVelocity ( Widget, XtPointer );
virtual void doOptionWhole ( Widget, XtPointer );

```

```
//---- Start editable code block: Bb protected
```

```
//---- End editable code block: Bb protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbUIResources[];
```

```
// Callbacks to interface with Motif
```

```

static void NextCallback ( Widget, XtPointer, XtPointer );
static void PrevCallback ( Widget, XtPointer, XtPointer );
static void doOption3DCallback ( Widget, XtPointer, XtPointer );
static void doOptionAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionColor2DCallback ( Widget, XtPointer, XtPointer );
static void doOptionGray2DCallback ( Widget, XtPointer, XtPointer );
static void doOptionMagnitudeCallback ( Widget, XtPointer, XtPointer );
static void doOptionNewAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionOtherCallback ( Widget, XtPointer, XtPointer );
static void doOptionPhaseCallback ( Widget, XtPointer, XtPointer );
static void doOptionROICallback ( Widget, XtPointer, XtPointer );
static void doOptionReferenceCallback ( Widget, XtPointer, XtPointer );
static void doOptionSimpleCallback ( Widget, XtPointer, XtPointer );
static void doOptionSplineCallback ( Widget, XtPointer, XtPointer );
static void doOptionStopAnimateCallback ( Widget, XtPointer, XtPointer );
static void doOptionVelocityCallback ( Widget, XtPointer, XtPointer );
static void doOptionWholeCallback ( Widget, XtPointer, XtPointer );

```

```
//---- Start editable code block: Bb private
```

```
//---- End editable code block: Bb private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for BbVelocity
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVelocityUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBVELOCITY_H
#define BBVELOCITY_H
#include "BbVelocityUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"

//---- End editable code block: headers and declarations

//---- BbVelocity class declaration

class BbVelocity : public BbVelocityUI
{
public:
    BbVelocity ( const char *, Widget );
    BbVelocity ( const char * );
    ~BbVelocity();
    const char * className();

    static VkComponent *CreateBbVelocity( const char *name, Widget parent );

//---- Start editable code block: BbVelocity public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

//---- End editable code block: BbVelocity public

protected:

    // These functions will be called as a result of callbacks
    // registered in BbVelocityUI

    virtual void Ratio ( Widget, XtPointer );
    virtual void doOption100 ( Widget, XtPointer );
    virtual void doOption25 ( Widget, XtPointer );

```

```
virtual void doOption58 ( Widget, XtPointer );
virtual void doOption7 ( Widget, XtPointer );
virtual void doOptionAsIs ( Widget, XtPointer );
virtual void doOptionFlowMasked ( Widget, XtPointer );
virtual void doOptionNone ( Widget, XtPointer );
virtual void doOptionROIMasked ( Widget, XtPointer );
virtual void threshMag ( Widget, XtPointer );
virtual void threshNeg ( Widget, XtPointer );
virtual void threshPos ( Widget, XtPointer );
```

```
//----- Start editable code block: BbVelocity protected
```

```
//----- End editable code block: BbVelocity protected
```

```
private:
```

```
static void* RegisterBbVelocityInterface();
```

```
//----- Start editable code block: BbVelocity private
```

```
//----- End editable code block: BbVelocity private
```

```
};
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```


User: meide
Host: phoenix
Class: phoenix
Job: BbRROIUI.h

733

```

/////////////////////////////////////////////////////////////////
//
// Header file for BbVelocityUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbVelocity is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbVelocity files
//
// Restrict changes to those sections between
// the "//--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
/////////////////////////////////////////////////////////////////
#ifndef BBVELOCITYUI_H
#define BBVELOCITYUI_H
#include <Vk/VkComponent.h>

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuItem;
class VkMenuItem;
class VkMenuItem;
class BbVelocityUI : public VkComponent
{
public:
    BbVelocityUI ( const char *, Widget );
    BbVelocityUI ( const char * );
    ~BbVelocityUI();
    void create ( Widget );
    const char * className();

    //---- Start editable code block: BbVelocity public

    //---- End editable code block: BbVelocity public

protected:

    // Widgets created by this class

    Widget _bbVelocity;
    Widget _labelMagThresh;
    Widget _labelNegThresh;
    Widget _labelPhase2Velocity;

```

```
Widget _labelPosThresh;
Widget _textfieldMagThresh;
Widget _textfieldNegThresh;
Widget _textfieldPosThresh;
Widget _textfieldRatio;
```

```
VkOptionMenu *_optionMenu4;
VkOptionMenu *_optionMenuVelocityMethod;
```

```
VkMenuItem *_option1;
VkMenuItem *_option100;
VkMenuItem *_option25;
VkMenuItem *_option75;
VkMenuItem *_optionAsIs;
VkMenuItem *_optionFlowMasked;
VkMenuItem *_optionNone;
VkMenuItem *_optionROIMasked;
```

```
// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions
```

```
virtual void Ratio ( Widget, XtPointer );
virtual void doOption100 ( Widget, XtPointer );
virtual void doOption25 ( Widget, XtPointer );
virtual void doOption50 ( Widget, XtPointer );
virtual void doOption75 ( Widget, XtPointer );
virtual void doOptionAsIs ( Widget, XtPointer );
virtual void doOptionFlowMasked ( Widget, XtPointer );
virtual void doOptionNone ( Widget, XtPointer );
virtual void doOptionROIMasked ( Widget, XtPointer );
virtual void threshMag ( Widget, XtPointer );
virtual void threshNeg ( Widget, XtPointer );
virtual void threshPos ( Widget, XtPointer );
```

```
//---- Start editable code block: BbVelocity protected
```

```
//---- End editable code block: BbVelocity protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultBbVelocityUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void RatioCallback ( Widget, XtPointer, XtPointer );
static void doOption100Callback ( Widget, XtPointer, XtPointer );
static void doOption25Callback ( Widget, XtPointer, XtPointer );
static void doOption50Callback ( Widget, XtPointer, XtPointer );
static void doOption75Callback ( Widget, XtPointer, XtPointer );
static void doOptionAsIsCallback ( Widget, XtPointer, XtPointer );
static void doOptionFlowMaskedCallback ( Widget, XtPointer, XtPointer );
static void doOptionNoneCallback ( Widget, XtPointer, XtPointer );
static void doOptionROIMaskedCallback ( Widget, XtPointer, XtPointer );
static void threshMagCallback ( Widget, XtPointer, XtPointer );
static void threshNegCallback ( Widget, XtPointer, XtPointer );
static void threshPosCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbVelocity private
```

```
    //---- End editable code block: BbVelocity private  
};  
//---- Start editable code block: End of generated code
```

736

```
//---- End editable code block: End of generated code  
#endif
```

```

////////////////////////////////////
//
// Header file for BbVisual
//
// This file is generated by RapidApp 1.2
//
// This class is derived from BbVisualUI which
// implements the user interface created in
// RapidApp. This class contains virtual
// functions that are called from the user interface.
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef BBVISUAL_H
#define BBVISUAL_H
#include "BbVisualUI.h"
//---- Start editable code block: headers and declarations

#include "ObjectManager.h"
#include <Xm/TextF.h>
#include <Vk/VkComponent.h>
#include <Xm/List.h>

//---- End editable code block: headers and declarations

//---- BbVisual class declaration

class BbVisual : public BbVisualUI
{
public:

    BbVisual ( const char *, Widget );
    BbVisual ( const char * );
    ~BbVisual();
    const char * className();

    static VkComponent *CreateBbVisual( const char *name, Widget parent );

    //---- Start editable code block: BbVisual public

    ObjectManager *_objMag;
    void set(ObjectManager *objMag) {_objMag = objMag;}

    void accept();
    void clear_vessel() {XmListDeleteAllItems(_scrolledListVessel4);}

    void set_flowdir(int);
    void add_flow(char *);

    char *get_patient() { return XmTextFieldGetString(_textfieldName);}
    char *get_vessel() { return XmTextFieldGetString(_textfieldVessel);}
    char *get_date() { return XmTextFieldGetString(_textfieldDate);}
    char *get_remark() { return XmTextFieldGetString(_textfieldRemark);}

    void set_info(char *name, char *studyDate, char *remark);

```

```

void setVessel(char *aomy);
void set_Path(char *p);
char *get_basePath();

//void update_histo(HistoView *his, int w, int h, short **img);

//---- End editable code block: BbVisual public

```

protected:

```

// These functions will be called as a result of callbacks
//.registered in BbVisualUI

```

```

virtual void Vessel ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doVeLICANeck ( Widget, XtPointer );
virtual void doVesBAdown ( Widget, XtPointer );
virtual void doVesBAup ( Widget, XtPointer );
virtual void doVesLACA ( Widget, XtPointer );
virtual void doVesLCCA ( Widget, XtPointer );
virtual void doVesLECA ( Widget, XtPointer );
virtual void doVesLICAItra ( Widget, XtPointer );
virtual void doVesLMCA ( Widget, XtPointer );
virtual void doVesLVA ( Widget, XtPointer );
virtual void doVesNew ( Widget, XtPointer );
virtual void doVesRACA ( Widget, XtPointer );
virtual void doVesRCCA ( Widget, XtPointer );
virtual void doVesRECA ( Widget, XtPointer );
virtual void doVesRICAItra ( Widget, XtPointer );
virtual void doVesRICANeck ( Widget, XtPointer );
virtual void doVesRMCA ( Widget, XtPointer );
virtual void doVesRVA ( Widget, XtPointer );
virtual void setToggleFlowNeg ( Widget, XtPointer );
virtual void setToggleFlowNeutral ( Widget, XtPointer );
virtual void setToggleFlowPos ( Widget, XtPointer );
virtual void userName ( Widget, XtPointer );
virtual void vesselRUser ( Widget, XtPointer );

```

```

//---- Start editable code block: BbVisual protected

```

```

//---- End editable code block: BbVisual protected

```

private:

```

static void* RegisterBbVisualInterface();

```

```

//---- Start editable code block: BbVisual private

```

```

//---- End editable code block: BbVisual private

```

```

};
//---- Start editable code block: End of generated code

```

```

//---- End editable code block: End of generated code

```

```

#endif

```



```

////////////////////////////////////
//
// Header file for BbVisualUI
//
// This file is generated by RapidApp 1.2
//
// This class implements the user interface portion of a class
// Normally it is not used directly.
// Instead the subclass, BbVisual is instantiated
//
// To extend or alter the behavior of this class, you should
// modify the BbVisual files
//
// Restrict changes to those sections between
// the "/*--- Start/End editable code block" markers
//
// This will allow RapidApp to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
//
////////////////////////////////////
#ifndef BBVISUALUI_H
#define BBVISUALUI_H
#include <Vk/VkComponent.h>

/*--- Start editable code block: headers and declarations

/*--- End editable code block: headers and declarations

class VkOptionMenu;
class VkMenuAction;
class VkMenuToggle;
class VkMenuItem;
class BbVisualUI : public VkComponent
{
public:

    BbVisualUI ( const char *, Widget );
    BbVisualUI ( const char * );
    ~BbVisualUI();
    void create ( Widget );
    const char * className();

    /*--- Start editable code block: BbVisual public

    /*--- End editable code block: BbVisual public

protected:

    // Widgets created by this class

    Widget _bbVisual;
    Widget _buttonAcceptUser;
    Widget _labelDate;
    Widget _labelDescription;

```



```

Widget _labelFlowDir;
Widget _labelVessel;
Widget _labelname;
Widget _radioboxFlowDir;
Widget _scrolledListVessel4;
Widget _scrolledWindow5;
Widget _textfieldDate;
Widget _textfieldName;
Widget _textfieldRemark;
Widget _textfieldVessel;
Widget _toggleFlowNeg;
Widget _toggleFlowNeutral;
Widget _toggleFlowPos;

```

```

VkOptionMenu *_optionMenu17;

```

```

VkMenuItem *_separator5;
VkMenuItem *_separator6;
VkMenuItem *_separator7;
VkMenuItem *_separator8;
VkMenuItem *_vesBAdown;
VkMenuItem *_vesBAup;
VkMenuItem *_vesLACA;
VkMenuItem *_vesLCCA;
VkMenuItem *_vesLECA;
VkMenuItem *_vesLICAINtra;
VkMenuItem *_vesLICANeck;
VkMenuItem *_vesLMCA;
VkMenuItem *_vesLVA;
VkMenuItem *_vesNew;
VkMenuItem *_vesRACA;
VkMenuItem *_vesRCCA;
VkMenuItem *_vesRECA;
VkMenuItem *_vesRICAINtra;
VkMenuItem *_vesRICANeck;
VkMenuItem *_vesRMCA;
VkMenuItem *_vesRVA;

```

```

// These virtual functions are called from the private callbacks (below)
// Intended to be overridden in derived classes to define actions

```

```

virtual void Vessel ( Widget, XtPointer );
virtual void doButtonAccept ( Widget, XtPointer );
virtual void doVeLICANeck ( Widget, XtPointer );
virtual void doVesBAdown ( Widget, XtPointer );
virtual void doVesBAup ( Widget, XtPointer );
virtual void doVesLACA ( Widget, XtPointer );
virtual void doVesLCCA ( Widget, XtPointer );
virtual void doVesLECA ( Widget, XtPointer );
virtual void doVesLICAINtra ( Widget, XtPointer );
virtual void doVesLMCA ( Widget, XtPointer );
virtual void doVesLVA ( Widget, XtPointer );
virtual void doVesNew ( Widget, XtPointer );
virtual void doVesRACA ( Widget, XtPointer );
virtual void doVesRCCA ( Widget, XtPointer );
virtual void doVesRECA ( Widget, XtPointer );
virtual void doVesRICAINtra ( Widget, XtPointer );
virtual void doVesRICANeck ( Widget, XtPointer );
virtual void doVesRMCA ( Widget, XtPointer );
virtual void doVesRVA ( Widget, XtPointer );
virtual void setToggleFlowNeg ( Widget, XtPointer );
virtual void setToggleFlowNeutral ( Widget, XtPointer );
virtual void setToggleFlowPos ( Widget, XtPointer );
virtual void userName ( Widget, XtPointer );
virtual void vesselRUser ( Widget, XtPointer );

```



```
//---- End editable code block: BbVisual protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultBbVisualUIResources[];
```

```
// Callbacks to interface with Motif
```

```
static void VesselCallback ( Widget, XtPointer, XtPointer );
static void doButtonAcceptCallback ( Widget, XtPointer, XtPointer );
static void doVeLICANeckCallback ( Widget, XtPointer, XtPointer );
static void doVesBADownCallback ( Widget, XtPointer, XtPointer );
static void doVesBAUpCallback ( Widget, XtPointer, XtPointer );
static void doVesLACACallback ( Widget, XtPointer, XtPointer );
static void doVesLCCACallback ( Widget, XtPointer, XtPointer );
static void doVesLECACallback ( Widget, XtPointer, XtPointer );
static void doVesLICAIntraCallback ( Widget, XtPointer, XtPointer );
static void doVesLMCACallback ( Widget, XtPointer, XtPointer );
static void doVesLVACallback ( Widget, XtPointer, XtPointer );
static void doVesNewCallback ( Widget, XtPointer, XtPointer );
static void doVesRACACallback ( Widget, XtPointer, XtPointer );
static void doVesRCCACallback ( Widget, XtPointer, XtPointer );
static void doVesRECACallback ( Widget, XtPointer, XtPointer );
static void doVesRICAItraCallback ( Widget, XtPointer, XtPointer );
static void doVesRICANeckCallback ( Widget, XtPointer, XtPointer );
static void doVesRMCACallback ( Widget, XtPointer, XtPointer );
static void doVesRVACallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowNegCallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowNeutralCallback ( Widget, XtPointer, XtPointer );
static void setToggleFlowPosCallback ( Widget, XtPointer, XtPointer );
static void userNameCallback ( Widget, XtPointer, XtPointer );
static void vesselRUserCallback ( Widget, XtPointer, XtPointer );
```

```
//---- Start editable code block: BbVisual private
```

```
//---- End editable code block: BbVisual private
```

```
};
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

/////////////////////////////////////////////////////////////////
//
// Header file for DeckLTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
//
// When you modify this header file, limit your changes to those
// areas between the "//---- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
/////////////////////////////////////////////////////////////////
#ifndef DECKLTABBEDDECK_H
#define DECKLTABBEDDECK_H
#include <Vk/VkTabbedDeck.h>
//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- DeckLTabbedDeck class declaration

class DeckLTabbedDeck : public VkTabbedDeck
{
public:
    DeckLTabbedDeck ( const char *, Widget );
    DeckLTabbedDeck ( const char * );
    ~DeckLTabbedDeck();
    const char * className();

    static VkComponent *CreateDeckLTabbedDeck( const char *name, Widget parent );

//---- Start editable code block: DeckLTabbedDeck public

void set(class ObjectManager *);

//---- End editable code block: DeckLTabbedDeck public

protected:

// Classes created by this class

class BbHistogram *_bbHistogram;
class BbDisplay *_bbDisplay;
class BbLROI *_bbLROI;
class BbLConfig *_bbLConfig;
class BbLPCMRA *_bbLPCMRA;
class BbLWaveform *_bbLWaveform;
class BbDetail *_bbDetail;
class BbLConfigNew *_bbLConfigNew;

// Widgets created by this class

```

```
//---- Start editable code block: DeckLTabbedDeck protected
```

```
//---- End editable code block: DeckLTabbedDeck protected
```

```
private:
```

```
// Array of default resources
```

```
static String      _defaultDeckLTabbedDeckResources[];
```

```
//---- Start editable code block: DeckLTabbedDeck private
```

```
//---- End editable code block: DeckLTabbedDeck private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

```

////////////////////////////////////
//
// Header file for DeckRTabbedDeck
//
// This file is generated by RapidApp 1.2
//
// This class is derived from VkTabbedDeck
//
// When you modify this header file, limit your changes to those
// areas between the "//----- Start/End editable code block" markers
//
// This will allow the builder to integrate changes more easily
//
// This class is a ViewKit user interface "component".
// For more information on how components are used, see the
// "ViewKit Programmers' Manual", and the RapidApp
// User's Guide.
////////////////////////////////////
#ifndef DECKRTABBEDDECK_H
#define DECKRTABBEDDECK_H
#include <Vk/VkTabbedDeck.h>
//----- Start editable code block: headers and declarations

//----- End editable code block: headers and declarations

//----- DeckRTabbedDeck class declaration

class DeckRTabbedDeck : public VkTabbedDeck
{
public:
    DeckRTabbedDeck ( const char *, Widget );
    DeckRTabbedDeck ( const char * );
    ~DeckRTabbedDeck();
    const char * className();

    static VkComponent *CreateDeckRTabbedDeck( const char *name, Widget parent );

//----- Start editable code block: DeckRTabbedDeck public

    void set(class ObjectManager *);

//----- End editable code block: DeckRTabbedDeck public

protected:

    // Classes created by this class

    class BbRHistogram *_bbRHistogram;
    class BbFlow *_bbFlow;
    class BbVisual *_bbVisual;
    class BbFormat *_bbFormat;
    class BbFlow3D *_bbFlow3D;
    class Bb3DLocalizer *_bb3DLocalizer;
    class BbVelocity *_bbVelocity;
    class Bb3D *_bb3D;
    class BbAnimation *_bbAnimation;
    class BbRROI *_bbRROI;

```

```
class BbRWaveform *_bbRWaveform;  
class BbRTable *_bbRTa
```

747

```
// Widgets created by this class
```

```
Widget _deckR;
```

```
//---- Start editable code block: DeckRTabbedDeck protected
```

```
//---- End editable code block: DeckRTabbedDeck protected
```

```
private:
```

```
// Array of default resources
```

```
static String _defaultDeckRTabbedDeckResources[];
```

```
//---- Start editable code block: DeckRTabbedDeck private
```

```
//---- End editable code block: DeckRTabbedDeck private
```

```
};  
//---- Start editable code block: End of generated code
```

```
//---- End editable code block: End of generated code
```

```
#endif
```

User: meide
Host: phoenix
Class: phoenix
Job: BbVelocityUI.h

748


```

////////////////////////////////////
//
// Header file for VkwindowMainWindow
//
// This class is a subclass of VkWindow
//
// Normally, very little in this file should need to be changed.
// Create/add/modify menus using RapidApp.
//
// Restrict changes to those sections between
// the "///--- Start/End editable code block" markers
// Doing so will allow you to make changes using RapidApp
// without losing any changes you may have made manually
//
////////////////////////////////////
#ifndef VKWINDOWMAINWINDOW_H
#define VKWINDOWMAINWINDOW_H
#include <Vk/VkWindow.h>

class VkMenuItem;
class VkMenuToggle;
class VkMenuConfirmFirstAction;
class VkSubMenu;
class VkRadioSubMenu;

//---- Start editable code block: headers and declarations

//---- End editable code block: headers and declarations

//---- VkwindowMainWindow class declaration

class VkwindowMainWindow: public VkWindow {
public:
    VkwindowMainWindow( const char * name,
                        ArgList args = NULL,
                        Cardinal argCount = 0 );
    ~VkwindowMainWindow();
    const char *className();
    virtual Boolean okToQuit();

    //---- Start editable code block: VkwindowMainWindow public
    //---- End editable code block: VkwindowMainWindow public

protected:

    // Classes created by this class

    class Bb *_bb;

    // Widgets created by this class

    // Menu items created by this class
    VkSubMenu *_filePane;
    VkMenuItem *_newButton;
    VkMenuItem *_openButton;

```

```

VkMenuItem *_saveButton;
VkMenuItem *_saveasButton;
VkMenuItem *_printButton;
VkMenuItem *_separator1;
VkMenuItem *_closeButton;
VkMenuItem *_exitButton;
VkSubMenu *_editPane;
VkMenuItem *_theVkUndoManagerButton;
VkMenuItem *_cutButton;
VkMenuItem *_copyButton;
VkMenuItem *_pasteButton;
VkSubMenu *_viewPane;
VkMenuItem *_imgInfo;
VkMenuItem *_mraInfo;
VkMenuItem *_pcmraLocalizer;
VkSubMenu *_user;
VkMenuItem *_novies;
VkMenuItem *_expert;

```

```
// Member functions called from callbacks
```

```

virtual void close ( Widget, XtPointer );
virtual void copy ( Widget, XtPointer );
virtual void cut ( Widget, XtPointer );
virtual void expertCallback ( Widget, XtPointer );
virtual void imgInfoCallback ( Widget, XtPointer );
virtual void mraInfoCallback ( Widget, XtPointer );
virtual void newFile ( Widget, XtPointer );
virtual void noviesCallback ( Widget, XtPointer );
virtual void openFile ( Widget, XtPointer );
virtual void paste ( Widget, XtPointer );
virtual void pcmraCutCallback ( Widget, XtPointer );
virtual void print ( Widget, XtPointer );
virtual void quit ( Widget, XtPointer );
virtual void save ( Widget, XtPointer );
virtual void saveas ( Widget, XtPointer );

```

```
//---- Start editable code block: VkwindowMainWindow protected
```

```
//---- End editable code block: VkwindowMainWindow protected
```

```
private:
```

```
// Callbacks to interface with Motif
```

```

static void closeCallback ( Widget, XtPointer, XtPointer );
static void copyCallback ( Widget, XtPointer, XtPointer );
static void cutCallback ( Widget, XtPointer, XtPointer );
static void expertCallbackCallback ( Widget, XtPointer, XtPointer );
static void imgInfoCallbackCallback ( Widget, XtPointer, XtPointer );
static void mraInfoCallbackCallback ( Widget, XtPointer, XtPointer );
static void newFileCallback ( Widget, XtPointer, XtPointer );
static void noviesCallbackCallback ( Widget, XtPointer, XtPointer );
static void openFileCallback ( Widget, XtPointer, XtPointer );
static void pasteCallback ( Widget, XtPointer, XtPointer );
static void pcmraCutCallbackCallback ( Widget, XtPointer, XtPointer );
static void printCallback ( Widget, XtPointer, XtPointer );
static void quitCallback ( Widget, XtPointer, XtPointer );
static void saveCallback ( Widget, XtPointer, XtPointer );
static void saveasCallback ( Widget, XtPointer, XtPointer );

```

```
static String _defaultVkwindowMainWindowResources[];
```

```
//----- Start editable code block: Vmainwindow private
```

```
//----- End editable code block: Vmainwindow private
```

```
};
```

```
//----- Start editable code block: End of generated code
```

```
//----- End editable code block: End of generated code
```

```
#endif
```

```
#ifndef MESSAGESRIGHT_H
#define MESSAGESRIGHT_H
```

752

```
#include "Flow.h"
#include "Points.h"
```

```
typedef struct {
```

```
    Boolean  show_status;
```

```
    int      img_select;  //ROI, Whole, Reference, Other
```

```
    int      img_type;    //CT, MR, PCMRA
```

```
    int      img_pcmra_type;  //MAGNITUDE, PHASE, VELOCITY
```

```
    int      img_anatomy;  //HEAD, NECK, BONE
```

```
    int      img_number;
```

```
    int      img_number_prev;
```

```
    float    img_zoom;
```

```
    int      img_scale_type;
```

```
    float    img_winCenter;
```

```
    float    img_winWidth;
```

```
    float    low, high;
```

```
    float    lowGrayWhole, highGrayWhole;
```

```
    float    lowGrayROI, highGrayROI;
```

```
    float    lowGrayRef, highGrayRef;
```

```
    float    lowGrayOther, highGrayOther;
```

```
    float    lowColorWhole, highColorWhole;
```

```
    float    lowColorROI, highColorROI;
```

```
    float    lowColorRef, highColorRef;
```

```
    float    lowColorOther, highColorOther;
```

```
    float    lowMagGrayWhole, highMagGrayWhole;
```

```
    float    lowMagGrayROI, highMagGrayROI;
```

```
    float    lowMagGrayRef, highMagGrayRef;
```

```
    float    lowMagGrayOther, highMagGrayOther;
```

```
    float    lowMagColorWhole, highMagColorWhole;
```

```
    float    lowMagColorROI, highMagColorROI;
```

```
    float    lowMagColorRef, highMagColorRef;
```

```
    float    lowMagColorOther, highMagColorOther;
```

```
    float    lowPhaGrayWhole, highPhaGrayWhole;
```

```
    float    lowPhaGrayROI, highPhaGrayROI;
```

```
    float    lowPhaGrayRef, highPhaGrayRef;
```

```
    float    lowPhaGrayOther, highPhaGrayOther;
```

```
    float    lowPhaColorWhole, highPhaColorWhole;
```

```
    float    lowPhaColorROI, highPhaColorROI;
```

```
    float    lowPhaColorRef, highPhaColorRef;
```

```
    float    lowPhaColorOther, highPhaColorOther;
```

```
    int      img_visual_type;  //GRAY, COLOR
```

```
    int      img_space;  //2D, 3D
```

```
    int      histo_status;
```

```
    float    histo_min;
```

```
    float    histo_max;
```

```
    int      roi_type;
```

```
    int      roi_action;
```

```
    int      roi_x, roi_y, roi_w, roi_h;
```

```
int      num_imgs;
int      num_cardiacs;

char     vesselName[100];
char     userName[100];
int      flowDir;
int      flowDir2;

Boolean  show_detail;

int      animate_mode;

int      velocity_select;
float    velocity_ratio;
int      low_magthresh;

int      flow_select;
int      flow_method;
FlowPara *flows;
int      flow_noiseLevel;

float    ratio3D;
int      camera;
float    YPos3D;
float    Height3D;
int      Fixed3D;
int      flow3D;

int      publish;
int      flow3DDir;
float    HR;

int      roi_changed;
unsigned char **roi_mask;
unsigned char **roi_flow;
unsigned char **roi_back;

Points   *roi_points;

char     pubDir[300];

} MessagesRight;

#endif
```

```
#ifndef POINT_H  
#define POINT_H
```

754

```
typedef struct {  
    float  x;  
    float  y;  
} Point;  
  
#endif
```

```
#ifndef ROI_STRUCT_H
#define ROI_STRUCT_H
#include <Vk/VkComponent.h>
#include "Points.h"

typedef struct {
    char    _name[100];
    Points  *_points;
} ROI_OBJ;

typedef struct {
    int      _numROIs;
    ROI_OBJ  _ROI_OBJ[10];
} ROI_Struct;

#endif
```

```
#ifndef STUDIES_H
#define STUDIES_H
```

```
typedef struct {
    char    name[300];
    int     start, end;
    int     start2, end2;
    char    ref[300];
} Study_Info;
```

```
typedef struct {
    int     num;
    Study_Info *study_info;
} Studies;
```

```
#endif
```



```
#ifndef MESSAGES_H
#define MESSAGES_H
```

756

```
typedef struct {
    char    img_dir[200];
    int     img_exam;
    int     img_series;
    int     img_start;
    int     img_end;
    char    img_type[10];
    char    img_ref[200];
    int     img_number;
    int     roi;
    int     roi_event;
} Messages;

#endif
```

```
#ifndef MESSAGESCREATED_H
#define MESSAGESCREATED_H
```

757

```
#include "Flow.h"
```

```
typedef struct {
    int    img_type;
    int    img_anatomy;
    int    img_number;
    float  img_winCenter;
    float  img_winWidth;
    float  img_RwinCenter;
    float  img_RwinWidth;
    float  img_zoom;
    int    img_pcmra_type;
    int    img_interpolation;
    int    img_visual_type;
    int    histo_status;
    float  histo_min;
    float  histo_max;
    int    roi_type;
    int    roi_event;
    int    roi_whole;
    int    roi_color;
    int    roi_show;
    int    proi_x, proi_y, proi_w, proi_h;

    int    num_imgs;
    FlowPara *flows;

    int    right_img_type;
    int    right_pcmra_type;

    int    rroi_changed;
    unsigned char **rroi_mask;
} MessagesCreated;

#endif
```

```
#ifndef MESSAGESLEFT_H
#define MESSAGESLEFT_H
```

758

```
typedef struct {
    int    img_type;
    int    img_pcmra_type;
    int    img_anatomy;

    int    img_number;

    int    user;
    int    layout;
    int    loc_x1, loc_y1, loc_x2, loc_y2;

    int    img_space;

    float  img_zoom;
    int    img_scale_type;

    int    img_zoom_select;

    int    flow_select;

    int    show_status;

    float  img_winCenter;
    float  img_winWidth;

    float  low, high;
    float  lowMag, highMag;
    float  lowPha, highPha;

    int    img_visual_type;

    int    histo_status;
    float  histo_min;
    float  histo_max;

    //Boolean roi_status;
    int    roi_mode;

    int    roi_type;
    int    roi_action;

    int    roi_x, roi_y, roi_w, roi_h;

    float  posThresh, negThresh, magThresh;
} MessagesLeft;

#endif
```

```
#ifndef MESSAGESLOADED_H  
#define MESSAGESLOADED_H
```

759

```
typedef struct {  
    char    img_dir[200];  
    int     img_exam;  
    int     img_series;  
    int     img_start;  
    int     img_end;  
    int     img_start2;  
    int     img_end2;  
    char    img_type[10];  
    char    img_anatomy[10];  
    char    img_ref[200];  
} MessagesLoaded;  
  
#endif
```

```

    // _imgViewLoc -> show();
    // ((DrawingArea *)_imgViewLoc) -> display(x0, y0, RIGHT_MAX_WIDTH, RIGHT_MAX_HEIGHT);
}
else
{
    _imgViewLoc = new ROIMedDrawingArea("GE", _bb->baseWidget(), 0);
    _imgViewLoc -> setObj(this);
    _imgViewLoc -> set(w, h, _img2->get_imgdata(), msgsRight.img_visual_type, msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.f);

    // _imgViewLoc -> show();
    // ((DrawingArea *)_imgViewLoc) -> display(xc-w2/2, yc-h2/2);
    // printf(" Right Origin:: %d %d\n", xc-w2/2, yc-h2/2);

}

_imgViewLoc -> _roi_type = msgsLeft.roi_type;
_imgViewLoc -> _roi_action = msgsLeft.roi_action;
}

void ObjectManager::update_Rhisto()
{
    if(msgsRight.histo_status == HISTOGRAM_COARSE)
        update_Rhisto1();
    else if(msgsRight.histo_status == HISTOGRAM_FINE)
        update_Rhisto2();
    else if(msgsRight.histo_status == HISTOGRAM_MAPPING)
        update_RhistoMapping();
    else if(msgsRight.histo_status == HISTOGRAM_ROI)
        update_RhistoROI();
}

void ObjectManager::update_Rhisto1()
{
    int w = _imgView2->get_width();
    int h = _imgView2->get_height();
    short **img = _imgView2->_zoomImg;

    int dw = 400;
    int dh = 80;

    if(_map != NULL) {delete _map; _map = NULL; _RHist -> remove_mapLabels();}

    if(_histoView2 == NULL)
    {
        _histoView2 = new HistoTwoLinesDrawingArea(dw, dh,
            "Rhisto", _RHist -> baseWidget() );
        _histoView2 -> set(_RHist->_labelLHistoMin2,
            _RHist->_labelLHistoMax2,
            _RHist->_labelLHistoLow2,
            _RHist->_labelLHistoHigh2);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(this, MY_RIGHT);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(w, h, img, dw);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> newTwoLines(msgsRight.img_winCenter,
            msgsRight.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> display(0, 35);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> show();
    }
    else
    {
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(w, h, img, dw);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> newTwoLines(msgsRight.img_winCenter,
            msgsRight.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> display();
        ((HistoTwoLinesDrawingArea *)_histoView2) -> _twolines -> draw();
    }
}

```

```

    }
}

void ObjectManager::update_Rhisto2()
{
    int w = _imgView2->get_width();
    int h = _imgView2->get_height();
    short **img = _imgView2->_zoomImg;

    int dw = 400;
    int dh = 80;

    if(_map != NULL) {delete _map; _map = NULL; _RHist -> remove_mapLabels();}

    if(_histoView2 == NULL)
    {
        _histoView2 = new HistoTwoLinesDrawingArea(dw, dh,
            "Rhisto", _RHist -> baseWidget() );
        _histoView2 -> set(_RHist->_labelLHistoMin2,
            _RHist->_labelLHistoMax2,
            _RHist->_labelLHistoLow2,
            _RHist->_labelLHistoHigh2);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(this, MY_RIGHT);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(w, h, img, dw, NULL,
            msgsRight.img_winCenter-200.0, msgsRight.img_winWidth+200.0);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> newTwoLines(msgsRight.img_winCenter
            msgsRight.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> display(0, 35);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> show();
    }
    else
    {
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(w, h, img, dw, NULL,
            msgsRight.img_winCenter-200.0, msgsRight.img_winWidth+200.0);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> newTwoLines(msgsRight.img_winCenter
            msgsRight.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> display();
        ((HistoTwoLinesDrawingArea *)_histoView2) -> _twolines -> draw();
    }
}

void ObjectManager::update_RhistoMapping()
{
    if(_map == NULL)
    {
        int w = 400;
        int h = 50;
        int i, j, tmp1, tmp2, tmp3;

        Utility_Math *um = new Utility_Math();
        float low = 0.0;
        float high = 300.0;

        float b1, b2;
        um -> lineParaFromTwoPoints(0.0, low-200.0, 50.0, low, &b1, &b2);
        float c1, c2;
        um -> lineParaFromTwoPoints(350.0, high, 400.0, high+168.0, &c1, &c2);

        short **mapImg = alloc_shimg(w, h);

        for(i=0; i<h; i++)
        {
            for(j=0; j<50; j++)
                mapImg[i][j] = um -> int_t(b1 * float(j) + b2);
        }
    }
}

```

```

    for(j=50; j<350; j++)
        mapImg[i][j] = j -

    for(j=350; j<w; j++)
        mapImg[i][j] = um -> int_t(c1 * float(j) + c2);
}

tmp1 = msgsRight.img_visual_type;
tmp2 = msgsRight.img_scale_type;
tmp3 = msgsRight.flowDir;

if(_histoView2 != NULL) {delete _histoView2; _histoView2 = NULL;}

_map = new MedDrawingArea("GE", _RHist -> baseWidget(), 0);
_map -> set(w, h, mapImg, tmp1, tmp2, 1.0, low, high, tmp3);

_map -> show();
((DrawingArea *)_map) -> display(0, 70);

_RHist -> create_mapLabels();

delete um;
free_shimg(mapImg);
}
_RHist -> set_mapLabels();
}

void ObjectManager::update_RhistROI()
{
    int w = _imgView2->get_width();
    int h = _imgView2->get_height();
    short **img = _imgView2->_zoomImg;

    int dw = 400;
    int dh = 80;

    if(_map != NULL) {delete _map; _map = NULL; _RHist -> remove_mapLabels();}

    if(_histoView2 != NULL && (_imgView2 -> _ROI) != NULL &&
        (_imgView2 -> _ROI -> _area) != NULL)
    {
        ((HistoTwoLinesDrawingArea *)_histoView2) -> set(w, h, img, dw,
            _imgView2 -> _ROI -> _area);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> newTwoLines(msgsRight.img_winCenter,
            msgsRight.img_winWidth);
        ((HistoTwoLinesDrawingArea *)_histoView2) -> display();
        ((HistoTwoLinesDrawingArea *)_histoView2) -> _twolines -> draw();
    }
}

int ObjectManager::get_tag(int num, float *x, float *xMin, float *xMax, float *avg)
{
    int i, j, k, tag;

    *xMin = x[0];
    *xMax = x[0];

    *avg = x[0];

    for(i=1; i<num; i++)
    {
        if(*xMin > x[i]) *xMin = x[i];
        if(*xMax < x[i]) *xMax = x[i];
        *avg += x[i];
    }
}

```

```
*avg /= float(num);
```

```
if(fabsf(*xMin) > fabsf(*xMax))
```

```
{
    float tmp = *xMax;
    *xMax = -(*xMin);
    *xMin = -tmp;
    tag = -1;
}
```

```
else tag = 1;
```

```
return tag;
```

```
}
```

```
void ObjectManager::update_Lwave(int in_vessel)
```

```
{
    int vessel = in_vessel;
    int num = _flow[vessel].numPoints;
    int size = num * msgsRight.num_cardiacs;
    int dw = 350;
    int dh = 130;
```

```
float *y0 = new float[num];
```

```
int i, j, k;
```

```
for(i=0; i<num; i++)
```

```
{
    switch (msgsLeft.flow_select)
    {
        case FLOW_VFR:
            y0[i] = _flow[vessel].vesselFlows[i].vfr;
            break;
        case FLOW_PSV:
            y0[i] = _flow[vessel].vesselFlows[i].psv;
            break;
        case FLOW_BSV:
            y0[i] = _flow[vessel].vesselFlows[i].bsv;
            break;
        case FLOW_MV:
            y0[i] = _flow[vessel].vesselFlows[i].mv;
            break;
        case FLOW_AREA:
            y0[i] = _flow[vessel].vesselFlows[i].area;
            break;
        default:
            break;
    }
}
```

```
float minI, maxI, avg;
```

```
int tag = get_tag(num, y0, &minI, &maxI, &avg);
```

```
avg *= tag;
```

```
switch (msgsLeft.flow_select)
```

```
{
    case FLOW_VFR:
        _LWave -> set_unit("mL/min");
        break;
    case FLOW_PSV:
        _LWave -> set_unit("cm/sec");
        break;
    case FLOW_BSV:
        _LWave -> set_unit("cm/sec");
        break;
}
```



```

    case FLOW_MV:
        _LWave -> set_unit("cm/sec");
        break;
    case FLOW_AREA:
        _LWave -> set_unit("cm^2");
        break;
    default:
        break;
}
_LWave -> set_info(minI, maxI, avg);

float *x = NULL;
float *y = new float[size];

k = 0;
for(j=0; j<msgsRight.num_cardiacs; j++)
for(i=0; i<num; i++)
{
    y[k] = tag * y0[i];
    k++;
}

delete y0;

if(_waveView == NULL)
{
    _waveView = new LineDrawingArea(dw, dh, "Lwave", _LWave -> baseWidget(), DRAW_CUF
    ((LineDrawingArea *)_waveView) -> set(size, x, y);
    ((LineDrawingArea *)_waveView) -> display(240, 10);
    ((LineDrawingArea *)_waveView) -> show();
}
else
{
    ((LineDrawingArea *)_waveView) -> set(size, x, y);
    ((LineDrawingArea *)_waveView) -> display();
}
}

void ObjectManager::update_Rwave(int in_vessel)
{
    int vessel = in_vessel;
    int num = _flow[vessel].numPoints;
    int size = num * msgsRight.num_cardiacs;
    int dw = 350;
    int dh = 130;

    float *y0 = new float[num];
    int i, j, k;

    for(i=0; i<num; i++)
    {
        switch (msgsRight.flow_select)
        {
            case FLOW_VFR:
                y0[i] = _flow[vessel].vesselFlows[i].vfr;
                break;
            case FLOW_PSV:
                y0[i] = _flow[vessel].vesselFlows[i].psv;
                break;
            case FLOW_BSV:
                y0[i] = _flow[vessel].vesselFlows[i].bsv;
                break;
            case FLOW_MV:
                y0[i] = _flow[vessel].vesselFlows[i].mv;

```

```

        break;
    case FLOW_AREA:
        y0[i] = _flow[vessel].vesselFlows[i].area;
        break;
    default:
        break;
}
}

float  minI, maxI, avg;
int tag = get_tag(num, y0, &minI, &maxI, &avg);
avg *= tag;
switch (msgsRight.flow_select)
{
    case FLOW_VFR:
        _RWave -> set_unit("mL/min");
        break;
    case FLOW_PSV:
        _RWave -> set_unit("cm/sec");
        break;
    case FLOW_BSV:
        _RWave -> set_unit("cm/sec");
        break;
    case FLOW_MV:
        _RWave -> set_unit("cm/sec");
        break;
    case FLOW_AREA:
        _RWave -> set_unit("cm^2");
        break;
    default:
        break;
}
_RWave -> set_info(minI, maxI, avg);

float *x = NULL;
float *y = new float[size];

k = 0;
for(j=0; j<msgsRight.num_cardiacs; j++)
for(i=0; i<num; i++)
{
    y[k] = tag * y0[i];
    k++;
}

delete y0;

if(_waveView2 == NULL)
{
    _waveView2 = new LineDrawingArea(dw, dh, "Lwave", _RWave -> baseWidget(), DRAW_CT
    ((LineDrawingArea *)_waveView2) -> set(size, x, y);
    ((LineDrawingArea *)_waveView2) -> display(230, 10);
    ((LineDrawingArea *)_waveView2) -> show();
}
else
{
    ((LineDrawingArea *)_waveView2) -> set(size, x, y);
    ((LineDrawingArea *)_waveView2) -> display();
}
}

void ObjectManager::remove_animate()
{
    if(_animate != NULL)
    {

```

```
XtRemoveTimeOut(_animate->_id);
```

766

```
if(_animate->_pixmap != NULL)
```

```
{
    XtReleaseGC(_animate->_widget, _animate->_gc);
    _animate->_gc = NULL;
```

```
for(int i=0; i<_animate->_num_imgs; i++)
```

```
{
    if(_animate->_pixmap[i] != NULL)
        XFreePixmap(XtDisplay(_animate->_widget), _animate->_pixmap[i]);
}
delete _animate->_pixmap;
```

```
if(_animate->_lWave != NULL && _animate->_gc != NULL)
    XtReleaseGC(_animate->_widget, _animate->_lWaveGC);
```

```
if(_animate->_rWave != NULL && _animate->_gc != NULL)
    XtReleaseGC(_animate->_widget, _animate->_rWaveGC);
```

```
remove_animate3D();
```

```
delete _animate;
_animate = NULL;
```

```
}
}

void ObjectManager::create_animate()
```

```
{
    if(_imgView2 == NULL || msgsRight.num_imgs < 2) return;
    if(_animate != NULL) remove_animate();
    if(progress != NULL) remove_progress();
```

```
_animate = new Animate;
```

```
_animate->_widget = NULL;
```

```
if(msgsRight.animate_mode == ANIMATE_1D)
```

```
{
    create_Lanimate1D();
    create_Ranimate1D();
```

```
_animate->_pixmap = NULL;
```

```
empty_animate3D();
```

```
if(_animate->_lWave != NULL || _animate->_rWave != NULL)
    start_animate();
```

```
}
else if(msgsRight.animate_mode == ANIMATE_2D)
```

```
{
    _animate->_toBeFinished = TRUE;
    create_animate2D();
    _animate->_lWave = NULL;
    _animate->_rWave = NULL;
    empty_animate3D();
```

```
}
else if(msgsRight.animate_mode == ANIMATE_3D)
```

```
{
    _animate->_toBeFinished = TRUE;
    create_Ranimate3D();
    _animate->_lWave = NULL;
    _animate->_rWave = NULL;
    _animate->_pixmap = NULL;
```

```
}
```

```

else if(msgsRight.anima node == ANIMATE_SYMPHONY)
{
    _animate -> _toBeFinished = TRUE;
    create_animateSymphony();
    create_Lanimate1D();
    create_Ranimate1D();
}
else
{
    delete _animate;
    _animate = NULL;
    return;
}
}

void ObjectManager::start_animate()
{
    float bpm = (float)(_img2 -> get_heart_rate());

    if(bpm <= 1 || bpm >= 200)
        bpm = (float)(_img -> get_heart_rate());

    if(bpm <= 1 || bpm >= 200) bpm = 80.0;
    printf("    start_animate  %f %d\n", bpm, _animate->_num_imgs);

    _animate -> _msec = (int)(60.0/bpm/(float)_animate->_num_imgs * 1000.0);

    _animate -> _img_number = 0;

    _animate -> _time_out = 1;
    _animate -> _firsttime = 1;

    animation();
}

void ObjectManager::create_Lanimate1D()
{
    if(_waveView != NULL)
    {
        _animate -> _lWave = _waveView;

        if(_animate->_widget == NULL)
        {
            _animate->_widget = ((LineDrawingArea *)_waveView) -> baseWidget();
            _animate->_num_imgs = msgsRight.num_imgs;
        }

        _animate -> _num_waves = ((LineDrawingArea *)_waveView) -> _size;
        _animate -> _wave_number = 0;

        Utility_Widget *uw = new Utility_Widget();

        _animate->_lWaveColor = COLOR_RED;
        _animate->_lWaveGC = uw -> get_GC(_animate->_widget, _animate->_lWaveColor);

        delete uw;
    }
    else
    {
        _animate -> _lWave = NULL;
    }
}

void ObjectManager::create_Ranimate1D()
{
    if(_waveView2 != NULL)

```

```

{
    _animate -> _rWave = _waveView2;

    if(_animate->_widget == NULL)
    {
        _animate->_widget = ((LineDrawingArea *)_waveView2) -> baseWidget();
        _animate->_num_imgs = msgsRight.num_imgs;
    }

    _animate -> _num_waves = ((LineDrawingArea *)_waveView2) -> _size;
    _animate -> _wave_number = 0;

    Utility_Widget *uw = new Utility_Widget();

    _animate->_rWaveColor = COLOR_RED;
    _animate->_rWaveGC = uw -> get_GC(_animate->_widget, _animate->_rWaveColor);

    delete uw;
}
else
{
    _animate -> _rWave = NULL;
}
}

void ObjectManager::create_animate2D()
{
    if(_animate->_widget == NULL )
    {
        _animate->_num_imgs = msgsRight.num_imgs;
        _animate->_widget = _imgView2 -> baseWidget();
    }

    _animate->_pixmap = new Pixmap[msgsRight.num_imgs];
    for(int i=0; i<msgsRight.num_imgs; i++)
        _animate->_pixmap[i] = NULL;

    _animate->_width = _imgView2 -> get_width();
    _animate->_height = _imgView2 -> get_height();

    _animate -> _gc = DefaultGCOFScreen(XtScreen(_animate->_widget));

    msgsRight.img_type = msgsLeft.img_type;
    msgsRight.img_pcmra_type = msgsLeft.img_pcmra_type;

    update_progress("Create 2D Images For Animation");

    Progress_Animate2D();
}

void ObjectManager::update_progress(char *title)
{
    if(progress == NULL )
    {
        progress = new Progress;
        progress -> window = NULL;
    }

    if(progress -> window == NULL)
    {
        progress -> window = new ProgressMainWindow("Progress Status");
        progress -> window -> init(title);
        progress -> window -> show();
        progress -> widget = progress -> window -> baseWidget();
    }
}

```

```

XMoveWindow(XtDisplay progress -> widget),
XtWindow(progress -> widget), 512, 512);

```

769

```

progress -> _objMag = this;
progress -> msec = 10;
}
else
{
progress -> window -> set_title(title);
}

```

```

progress -> curr = msgsLoaded.img_start;
progress -> firsttime = 1;
progress -> time_out = 1;
}

```

```

void ObjectManager::remove_progress()

```

```

{
if(progress != NULL)
{
if(progress -> window != NULL)
{
delete progress -> window;
progress -> window = NULL;
}
delete progress;
progress = NULL;
}
}

```

```

void ObjectManager::update_flow()

```

```

{
//
// Flow
//
Utility_Math *um = new Utility_Math();
Utility_Widget *uw = new Utility_Widget();
Utility_Vision *uv = new Utility_Vision();
Utility *u = new Utility();

if(msgsRight.img_type == IMAGE_PCMRA && msgsRight.img_select == RIGHT_IMG_ROI)
{
unsigned char **area_flow = msgsRight.roi_flow;

//if(_imgView2 -> _ROI != NULL && _imgView2 -> _ROI -> _area != NULL)
if(area_flow != NULL)
{
int w1 = _img2 -> get_width();
int h1 = _img2 -> get_height();
short **img1;
GE_PCMRA_HEADER_OBJ *pc;

if(msgsRight.img_pcmra_type == PCMRA_VELOCITY)
{
img1 = _img2 -> get_imgdata();
pc = _img->get_header();
}
else if(msgsRight.img_pcmra_type == PCMRA_MAGNITUDE)
{
char fname[300];
int tmp = msgsRight.img_number;
sprintf(fname, "%s/E%dS%dI%d.MR", msgsLoaded.img_dir, msgsLoaded.img_ex,
msgsLoaded.img_series, tmp);

ImgGE *imgGE = new ImgGE(fname);
printf(" Flow::PCMRA_MAGNITUDE fname=%s\n", fname);
}
}
}

```

```

if(_imgView2 != NULL)
{
    p = get_RscaleSize(_imgView2 -> _zoom, &w2, &h2);
    c = get_RscaleSize(msgsRight.img_zoom, &w1, &h1);
}

printf(" update_RimgView:: %d %d  %f\n", _img2->get_width(), _img2->get_height(), n

if(_imgView2 != NULL && ( (!c && !p) || (p && c && w1 <= w2 && h1 <= h2) ) )
{
    //
    // The imgsize is under control
    //
    _imgView2 -> set(_img2->get_width(), _img2->get_height(), _img2->get_imgdata(),
        msgsRight.img_visual_type, msgsRight.img_scale_type,
        msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.

    int xc = 926;
    int yc = 346;
    ((DrawingArea *)_imgView2) -> set_Origin(xc - int(float(w1)/2.0), yc - int(float(
    printf(" Right Origin:: %d %d\n", xc - int(float(w1)/2.0), yc - int(float(h1)/2.0
    _imgView2 ->display();

    if(_imgViewLoc != NULL && msgsRight.img_select == RIGHT_IMG_REF)
        _imgViewLoc -> set(_img2->get_width(), _img2->get_height(), _img2->get_imgdata(
        msgsRight.img_visual_type, msgsRight.img_scale_type,
        msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.
    else if(_imgViewLoc == NULL && msgsRight.img_select == RIGHT_IMG_REF)
        new_RimgViewLoc();
}
else
{
    if(_imgView2 != NULL)
    {
        if(_animate != NULL) remove_animate();
        delete _imgView2;
    }
    new_RimgView();
    if(msgsRight.img_select == RIGHT_IMG_REF) new_RimgViewLoc();
}

//display_ROI();
//if(msgsRight.img_select == RIGHT_IMG_REF) _imgViewLoc = _imgView2;
if(_imgView2 -> _zoomImg != NULL) update_Rhisto();

}

void ObjectManager::display_ROI()
{
    _imgView2 -> EraseROI();
    int i = msgsRight.img_number - msgsLoaded.img_start;
    if(_ROIS->_ROI[i]._numROIs > 0)
        ((BbRROI *)_RROI) -> draw_AllROI(i);
    else if(msgsRight.img_number_prev != -1)
    {
        int j = msgsRight.img_number_prev - msgsLoaded.img_start;

        printf("display_ROI  prev: %d    curr: %d\n", msgsRight.img_number,
            msgsRight.img_number_prev);

        if(_ROIS->_ROI[j]._numROIs > 0)
        {
            ((BbRROI *)_RROI) -> draw_AllROI(j);
            ((BbRROI *)_RROI) -> add_AllROI(i, j);
        }
    }
}

```

```

}

void ObjectManager::update_RimgView(float center, float width)
{
    if(msgsRight.img_space == IMAGE_2D)
    {
        update_Rlowhigh();
        _imgView2 -> _flowDir = msgsRight.flowDir;

        _imgView2 -> update(center, width);
        if(_histoView2 != NULL)
        {
            ((HistoTwoLinesDrawingArea *)_histoView2) -> update_lowhigh(center, width);
            ((HistoTwoLinesDrawingArea *)_histoView2) -> change();
        }
    }
}

void ObjectManager::update_RimgView(int scale_method)
{
    if(msgsRight.img_space == IMAGE_2D)
    {
        _imgView2 -> update(scale_method);
        if(_imgView2 -> _zoomImg != NULL) update_Rhisto();
    }
}

void ObjectManager::update_Rvisual(int visual_method)
{
    hide3D();
    show2D();

    set_Rlowhigh();
    _imgView2 -> _winCenter = msgsRight.img_winCenter;
    _imgView2 -> _winWidth = msgsRight.img_winWidth;

    _imgView2 -> update_visual(visual_method);
    if(_imgView2 -> _zoomImg != NULL) update_Rhisto();
}

Boolean ObjectManager::get_RscaleSize(float zoom, int *w, int *h)
{
    int w1 = _img2->get_width();
    int h1 = _img2->get_height();

    Utility_Math *u = new Utility_Math();

    *w = u->int_t(w1 * zoom);
    *h = u->int_t(h1 * zoom);

    delete u;

    if(*w > RIGHT_MAX_WIDTH || *h > RIGHT_MAX_HEIGHT) return TRUE;
    else return FALSE;
}

void ObjectManager::new_RimgView()
{
    int w = _img2->get_width();
    int h = _img2->get_height();

    int w2, w3;
    int h2, h3;
    int x0, y0, xc, yc;

```



```

if(get_RscaleSize(msgsRight.img_zoom, &w2, &h2))
{
    if(w2 > RIGHT_MAX_WIDTH) w3 = RIGHT_MAX_WIDTH;
    else w3 = w2;
    if(h2 > RIGHT_MAX_HEIGHT) h3 = RIGHT_MAX_HEIGHT;
    else h3 = h2;
}

xc = 926;
yc = 346;

if(w3 > w2) x0 = 670;
else x0 = 670 + (RIGHT_MAX_WIDTH - w3)/2;

if(h3 > h2) y0 = 90;
else y0 = 90 + (RIGHT_MAX_HEIGHT - h3)/2;

if(get_RscaleSize(msgsRight.img_zoom, &w2, &h2))
{
    _imgView2 = new ROIMedDrawingArea("GE", _bb->baseWidget(), 1);
    _imgView2 -> setObj(this);
    _imgView2 -> set(w, h, _img2->get_imgdata(), msgsRight.img_visual_type, msgsRight.
    msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.f
    _imgView2 -> show();
    ((DrawingArea *)_imgView2) -> display(x0, y0, w3, h3);
}
else
{
    _imgView2 = new ROIMedDrawingArea("GE", _bb->baseWidget(), 0);
    _imgView2 -> setObj(this);
    _imgView2 -> set(w, h, _img2->get_imgdata(), msgsRight.img_visual_type, msgsRight
    msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.f
    _imgView2 -> show();
    ((DrawingArea *)_imgView2) -> display(xc-w2/2, yc-h2/2);
    printf(" Right Origin:: %d %d\n", xc-w2/2, yc-h2/2);
}

_imgView2 -> _roi_type = msgsLeft.roi_type;
_imgView2 -> _roi_action = msgsLeft.roi_action;
}

void ObjectManager::new_RimgViewLoc()
{
    if(_imgViewLoc != NULL)
    {
        delete _imgViewLoc;
        _imgViewLoc = NULL;
    }

    int w = _img2->get_width();
    int h = _img2->get_height();

    int w2;
    int h2;

    int xc = 316;
    int yc = 346+128;
    int x0 = 60;
    int y0 = 346;

    if(get_RscaleSize(msgsRight.img_zoom, &w2, &h2))
    {
        _imgViewLoc = new ROIMedDrawingArea("GE", _bb->baseWidget(), 1);
        _imgViewLoc -> setObj(this);
        _imgViewLoc -> set(w, h, _img2->get_imgdata(), msgsRight.img_visual_type, msgsRi
        msgsRight.img_zoom, msgsRight.img_winCenter, msgsRight.img_winWidth, msgsRight.f

```